Department of Computer Science

Technical Report

Undergraduate Dissertation: Multi-agent simulation of the dynamics of social exclusion in school choice

Perdita Robinson

Technical Report 2007-14 **June 2007**

ISSN 1740-9497

Contact Address:
Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom
URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

Multi-agent simulation of the dynamics of social exclusion in school choice

Perdita Robinson

BSc in Computer Science
University of Bath
May 2007

MULTI-AGENT SIMULATION OF THE DYNAMICS OF SOCIAL EXCLUSION IN SCHOOL CHOICE

Submitted by Perdita Robinson

COPYRIGHT

DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signature ……

Signature ……

Abstract

The phenomenon of school competition for middle-class children has been widely publicised as causing social inequity, as the more successful schools are led to exclude working-class children, who are not as profitable as their middle-class peers (Jordan, 1996). Room and Britton (2006b) constructed a mathematical model of this process, investigating the "catastrophic downward trajectories" taken by schools failing to attract middle-class children. However, the limitations of equation-based modelling meant they were restricted to considering only two competing schools, and to including only the most basic factors of school choice. This project moved from their initial macro-level view of the problem domain to a micro-analysis of the individual behaviours leading to the inequitable outcome. The introduction of a production rule system for each agent caused severe performance problems and limited the number of simulations that could be run. Nonetheless, the agent-based approach gave us new insights into the area that enabled us to suggest an amendment to the original model. We were also able to replicate Room and Britton's original results, and discovered that the fundamental factors they use determine an intrinsic social inequity that cannot be overcome by external influences.

Acknowledgements

With many thanks to my supervisor, Dr Julian Padget, for innumerable hours of discussion; and also to Prof. Graham Room, whose advice and suggestions have been invaluable.

I would also like to thank Elaine and Lynne for their proofreading, and everyone who gave me computer time, especially the Department of Computer Science for the use of alis.

Table of Contents

Table of Figures

1 Introduction

Room and Britton (2006b) created a mathematical model to investigate competition between secondary schools for middle-class children. This is widely known to lead to social inequity, with the more successful schools excluding working-class children (Davies, 1999). Due to the complexities of equation-based modelling, Room and Britton were restricted to considering only two competing schools, and to including only the basic factors of school admissions policies and parental preferences for successful schools. These preferences are argued to be stronger for middle-class parents, which serves to reinforce the social exclusion effect. However, in real life, many more factors come into play that we would now like to consider. We are also interested in moving from their initial macro-level view of the problem domain to a micro-analysis of the individual behaviours leading to the inequitable outcome. This would allow us to see how small changes in behaviour and in policy could trickle through to affect the macro outcome, and to deepen our understanding of the system in the process.

A multi-agent model seems ideally suited to this purpose. We believe that it will be relatively straightforward to convert Room and Britton's model into an agent-based version that is easily extended to incorporate new behaviours and factors of interest. In the process, we hope to gain new insights into the underlying processes of social exclusion.

1.1 Aim

The aim of this project was to develop and analyse a multi-agent simulation of the dynamics of social exclusion in the domain of school choice, initially equivalent to Room and Britton's equation-based model, and subsequently incorporating some extensions to demonstrate the advantages of agent-based modelling.

1.2 Objectives

1. Produce an agent-based model that is equivalent to Room and Britton's model, in that it produces the same two equilibrium states under the same conditions
2. Evaluate the suitability of the technology used for the initial prototype, and choose another framework for a more flexible model iteration if necessary
3. Develop a second, more flexible model (if necessary) and experiment with possible refinements or extensions, including agent heterogeneity.
4. Analyse sets of simulation runs using appropriate statistical techniques. Analysis should focus on the dynamics of the system, i.e. the underlying behaviours and processes.
5. Reflect on the advantages of agent-based modelling compared to the original model.

2 Literature Review

2.1 Socio-political background

The Baker school reforms of the 1980s, beginning with the 1988 Education Reform Act, have been widely criticised (e.g. Davies (1999); Karsten et al. (2001); West and Pennell (2002)). The reforms radically transformed the British education system, transferring control over which school a child would attend from the education authorities to parents, and making school performance data publicly available, allowing the media to rank schools in 'league tables'. Parents relied on these rankings heavily when choosing a school for their child (Davies, 1999). In addition, budgets have been delegated down to individual schools, making each school responsible for managing its own resources.

The effect of these policy changes was to create a market-based system of "competitive assortative mating" (Room and Britton, 2006a). This describes a reciprocal form of competition where parents wish to send their children to the best-rated schools, while schools attempt to attract those pupils who bring with them the best net profit. Schools are encouraged to recruit children whose contribution in terms of state funding meets, at a minimum, the expense of educating them. These are the middle-class children, who tend to perform better at school (thereby boosting the school's league table ranking) and are less likely to be expensive 'problem children' (Jordan, 1996) who bring with them no more funding than other children. This social divide is aggravated by the greater ability of middle-class families to move to a better school's catchment area. "All parents could choose, but some could choose more than others" (Davies, 1999). In addition, working-class parents are thought by Room and Britton to target these high-rated schools to a lesser degree than middle-class parents, perhaps because of placing a lower priority on quality of education, or out of a concern that the child would feel out of place surrounded by middle-class peers.

This overall effect of class polarisation is commonly referred to as 'social exclusion'. Perhaps because it refers to such a politically fraught issue, this term has no single standard definition. The Oxford Dictionary of Sociology (Scott and Marshall, 2005) identifies three main meanings; in this context, it shall be used in the sense of Room (1999), who talks of community-wide disadvantage in areas of social participation and integration, as opposed to focusing on a narrow income-based equivalent to the old concept of 'poverty', considering each household separately. This is closest in meaning to Scott and Marshall's concept of "social rights and [..] the barriers or processes by which people are prevented from exercising these", the social right in question being the right to a good education, in this instance.

2.2 Existing research

Previous work by Room and Britton (2006a) has investigated this market-based system by considering the interactions between institutional and household strategies. They take a quantitative system dynamics approach to the problem by modelling "runaway loops" in the system using differential equations. This involves identifying all influences amongst the system's attributes, and finding those subsystems where the direction of influence feeds back on itself in a self-reinforcing loop. These are the areas of interest to them. Their model shows how an initial equity state of two schools with equal numbers of pupils can easily be unbalanced if one school is perceived to be better than the other,

leading inexorably to an "inequity state" where the better school excels at the expense of the other one, and a class polarisation emerges.

Room and Britton adapt Schelling's (1978) classic racial segregation model of "tolerance schedules" that describe willingness to live in an area depending on the ratio of black to white neighbours, to "preference schedules" that indicate parents' preference for schools with a particular fraction of middle-class children. This preference is taken to be more pronounced in middle-class families for a variety of possible reasons, as discussed in section 2.1.

The study considered two types of school strategy: either the school allocates places on a social class-blind basis, or it gives preference to middle-class children. The latter is the strategy believed to be encouraged by the Baker reforms. Room and Britton found that it tended to produce the class polarisation effect more rapidly and pronouncedly.

Parental strategies were also two-fold in the study. This took the form of varying the strength of parents' preference for middle-class schools (for parents from both classes). The model's key finding concerns the interaction of these two strategies: those of the schools and those of the parents. When both parental preference for middle-class schools, and school preference for middle-class parents were strong, as in the current socio-political climate, only a small increase or decrease in a school's reputation led to the polarised inequity state with great likelihood and stability. If, however, only one of those conditions held, the other had to be very pronounced in order to achieve even a partial polarisation. A way of interpreting these results is to view the development of these strategies as a co-evolutionary process; to understand this, we must first visit the domain of complex systems.

2.3 Complex systems

Flake (1998) considers a system to be 'complex' if it consists of individual components whose interactions give rise to new, 'emergent' properties of the system as a whole that cannot be ascribed to the components themselves. Could the market-based school system be a complex system? Room and Britton (2006a) do not explicitly argue that it is as they do not use this terminology, but present some strong arguments in their paper that support this hypothesis. The class polarisation effect that can be observed in the system is presented as an emergent property: despite the simplicity inherent in the parents' and schools' actions, the cumulative effect of their interactions would "hardly have been possible" to foresee. In a complex system, it is impossible to make global inferences based purely on data about local choices (Albin, 1998).

But more than this, Room and Briton investigate an attribute often ascribed to complex systems: self-organisation. A self-organising system is one whose elements can "spontaneously" organise themselves into new patterns and behaviours that were not designed by any individual (Mitleton-Kelly, 1997). They hypothesise that class polarisation was already present in the system to some degree before the Baker reforms, but that the reforms then acted as the necessary trigger for self-organisation to exacerbate it, drawing a parallel to "control variables" in a complex system that determine whether self-organisation can occur.

Finally, the mathematical model they construct is a non-linear set of equations for modelling system dynamics. Nonlinear dynamics form the foundation of complex systems, and especially complex adaptive systems (Albin, 1998), which we shall examine next.

Complex adaptive systems (CAS hereafter) are complex systems whose components, or actors, are reactive: they can adapt to a changing environment (Albin, 1998). We could

say that our system is a CAS in which each actor, for example head teachers as the representatives of schools, and parents as the representatives of their households, adapts to an environment that consists of current legislation and government policies regarding school choice, as well as the responses of all the other actors.

All the actors' strategies could be said to be 'co-evolving' with each other. Here complex systems borrow a term from biology that some biologists feel is over-used even in the field of biology itself (e.g. Janzen, 1980), having become a synonym for 'interaction' and 'symbiosis'. It seems appropriate because there is a true evolutionary response in one population (e.g. head teachers) to a trait of the other (parents), followed by a response by the parents of adapting their own strategy again, and so on, rather than a mere adaptation of actors to their environment. ('Evolutionary' is used here to refer to an evolving strategy, much like in the field of genetic algorithms). Again, Room and Britton (2006a) do not explicitly use the term 'co-evolution', but explain that, as well as institutions (schools) shaping the fate of households through social exclusion, "households themselves pursue strategies vis-à-vis these various institutions: shaping them, resisting them, bypassing them, extracting benefit from their operations."

2.3.1 Complex systems in sociology

The status of the market system as a CAS can, however, be contested, depending on which definition of a CAS one uses. Vidgen and Wang (2004) view non-linearity as involving a series of positive feedbacks that can potentially push the system far along a trajectory. So far, their definition is in correspondence with our system moving towards class polarisation. However, the trajectory they mean is one that moves away from equilibrium, whereas Room and Britton's model is clearly pushed towards an equilibrium; that of the 'inequity state'. They draw on literature that emphasises complexity as taking place on "the edge of chaos", a region in which the long-term trajectory of the system is fundamentally unpredictable. However, we have satisfactorily established that the school market system meets ***a*** definition of a CAS, even if it does not match everyone's.

Other researchers question whether it is appropriate to use the generic concept of CAS in a social system, as the social sciences may differ considerably from 'hard' sciences like physics. Vidgen and Wang (2004) warn that the application of terminology should be done in the sense of making a metaphor or an analogy, rather than assuming that proofs from mathematics and physics translate directly. Mitleton-Kelly (1997) agrees, suggesting that generic CAS theory is a useful starting point for the study of complex social systems, but that it will need to be adapted. Albin (1998) believes that it is acceptable to view a social system as a CAS, provided the researcher is aware that the interactions are taking place between conscious beings. The nature of human consciousness is outside the scope of this project, but it is worth bearing in mind that it may affect the validity of parallels drawn with generic CAS.

Perhaps a more important question than whether a social system is a CAS is whether it is *useful* to categorise it as a complex system.

McIntyre (1998) questions the validity of the entire concept of complex systems. He proposes that what we perceive as complexity is not truly due to the real nature of the system, but rather stems from our own inability to comprehend the world around us at a deeper level. It cannot be denied that there are limits to human knowledge and understanding. But if complex systems really are just an artefact of these limits, then perhaps they are just as useful a concept to us as if they were a 'real' phenomenon, because we can still use them as a way of furthering our understanding. They could be viewed as a 'crutch' on our path to understanding this new level of existence, much like the manner in which children are presented with simplified theories of physics at school.

Albin (1998), however, is concerned about the opposite possibility: that we might not be able to perceive the complexity inherent in social and economic systems, and will therefore assume that it is not present. He emphasises the importance of modelling such systems through a less complex "surrogate" system, in order to build our understanding of what we cannot represent directly. Similarly, although arguing from a different starting point, McIntyre suggests that the usefulness in labelling a system as 'complex' is to recognise that our explanation of it is lacking, and so to attempt to find a simpler one, which is just as valid.

Vicsek (2002) sees the importance of complexity theory as our realisation that "the laws of the whole cannot be deduced by digging into the details". We shall return to this point later when we consider the motivation for an agent-based model of the system, which will simulate this emergence of properties of the whole from the interactions of its parts without using deduction to derive them. Gilbert (2004) suggests that nearly all interesting features of human societies may be emergent, due to the many non-linear interactions people engage in.

2.4 Social simulations

Now that we have decided to view the market-based school system as a complex system, the motivation for this project becomes clear: to develop a more sophisticated model than the original differential equations, and to use it to investigate the underlying processes of emergence and co-evolution in the system. A computer simulation seems ideally suited to this purpose. It can be executed under various conditions of interest, producing results not only on the eventual outcome but also concerning the internal processes and trajectories. In contrast, Room and Britton's earlier model was analysed structurally in order to draw conclusions about its equilibria.

Most social simulation researchers adopt Ostrom's suggestion (1988) that computer simulation is a "third way" of doing research, supplementing the existing techniques of verbal argumentation and formal representation through equations (e.g. Gilbert and Terna, 2000). This new "symbol system" is more tractable than mathematical modelling, and more rigorous, albeit less refined, than natural language. Hypotheses about processes can be examined experimentally, and emergence can be observed directly (Brenner and Werker, 2006). This is especially useful when natural or even laboratory experiments cannot be carried out, such as in matters of social policy,

To model a "target" system, an abstraction of it must be obtained, preferably motivated by theory (Gilbert and Terna, 2000), although a valid use of simulation is also as a "thought experiment" that explores a particular abstraction. We shall use Room and Britton's model (2006a; 2006b) and related theories as a basis for our abstraction. Any reasonably complicated system probably has an infinite number of possible models, but some are more appropriate than others, depending on the research aim (Gulyás, 2005; Gilbert and Doran, 1994). Since our goal is to build on their work, aiming foremost towards greater understanding but perhaps also towards prediction, their model is the best starting point for ours. This model also has the advantage of being very simple, so that we need only add complicating features if they are genuinely needed, rather than inadvertently designing a model with unnecessary components from the beginning.

We should consider that prediction may be too high an aim. Gilbert (1995) argues that the assumption that the best test of a theory is that it predicts successfully is not appropriate for non-linear systems, because even once their behaviour is understood, it is still impossible to predict. Batten (2000, p.259) does not see prediction as the primary purpose of simulation; he says that "inability to predict can be soothed by a growing ability to adapt and coevolve harmoniously – just like we find in nature". There is also concern

stemming from the debate of whether sociology is a science; if it is not, then it may be asserted that it should not attempt prediction (Henshel, 1982). Henshel also references the long-standing position that social forecasting is "immoral" because it removes people's motivation to try to control their own destiny. However, Room and Britton (2006a) hold that it would not have been possible to predict the effects of the Baker reforms without modelling the processes, and social policy is clearly an important area in which accurate predictions can lead towards better decisions.

It is commonly thought that the aim of a social simulation model should be either towards prediction, or towards understanding (e.g. Gilbert and Troitzsch, 1999, p.4). There is some methodological conflict in aiming towards both; Gulyás (2005) argues that predictive models require as much detail as possible to increase accuracy, while models aiming towards understanding, so-called "thought experiments", might try to be more general. In addition, prediction requires a high degree of quantitative validation against empirical data, while thought experiments allow for qualitative validation. However, we could argue that the model will be more thoroughly validated if both forms of validation are used, so there is no real conflict. It might also be useful to have the rigour demanded by prediction imposed on us, to encourage precision of thought (Henshel, 1982).

Social simulations are a comparatively new field of interest, compared to, for example, simulations in economics or the natural sciences. When simulation technology was first developed in the 1950s, it was not sophisticated enough to model realistic human societies. Instead, the power of computers was simply harnessed directly to solve mathematical models (Conte et al., 1998). Interest from social scientists faded and the field was quiet for around 25 years (Mosler, 2000). Recent developments in distributed artificial intelligence now overcome this barrier, introducing the concept of intelligent 'agents' who can stand in for humans in simulations (Gilbert and Doran, 1994, p. vii). Gilbert and Terna (2000) suggest that this need for especially realistic models is due to the main value of simulation for social scientists lying in theory development, rather than prediction, so the underlying mechanics are often more important than the eventual outcome.

2.4.1 Macrosimulation

The most basic type of social simulation is macrosimulation, which was developed in the '60s. It typically uses sets of differential equations to predict demographic variables based on feedback processes (Macy and Willer, 2002). This is essentially what Room and Britton's systems dynamics model does. As a top-down approach, macrosimulation is useful for gaining an appreciation of the overall principles at work, but to gain a deeper understanding, it is useful to examine the smaller units behind the macro effects using a bottom-up approach such as microsimulation.

2.4.2 Microsimulation and Markov processes

Microsimulation was developed in the '70s for analysing the possible effects of social policy changes on a society (Gilbert, 1995). In other words, it is aimed towards prediction, which ties in with Gilbert and Terna's theory that prediction is all that was feasible in early simulations. A microsimulation starts with a snapshot of sample population data from a certain point in time, and then iteratively simulates the effect of one year passing, examining each individual in turn. Demographic probabilities are used, e.g. the likelihood of a man aged 70-80 dying in any given year.

Although a population-wide effect (for example, increased demand on state pension funds) is built up from individual circumstances, this effect should not be termed emergence, because it simply consists of aggregate data, and its origin can therefore be

deduced logically (e.g. if pensioners live longer, more people will be drawing on the state pension fund). In fact, microsimulation does not represent a very significant advance on macrosimulation as it is still oriented towards forecasting macro effects, although it does allow policies to be considered that are aimed at altering individual behaviour (Macy and Willer, 2002). Gulyás (2005) calls microsimulation the "middle step" between macrosimulation and agent-based simulation.

He also identifies another type of model, Markov processes, which work on the macro level but incorporate stochastic elements like microsimulation does, and are therefore also closer to agent-based modelling than macrosimulation. However, they are only practical for systems with low complexity, and are therefore not suitable for representing the school market system; furthermore, we wish to examine the micro-macro link, as discussed later (see section 2.5.2).

The major drawback of the microsimulation approach, aside from its simplicity (which could also be considered a virtue, as discussed later) is that it considers each individual in complete isolation, without interaction; nor, like macrosimulation, does it allow for adaptation. This means that it can give no insight into the underlying processes that generated the data. Because of this, microsimulation tends to be used only for prediction, not for improving our understanding of a social phenomenon (Gilbert, 1995).

2.4.3 Cellular automata (CA)

Simulations based on CA address our criticism of microsimulation by modelling interactions between individuals. Schelling's racial segregation model (1978) is the canonical CA model. The other prominent model is Latané's much more recent model of social impact, which investigates the effects of social influence on attitude changes in a society (e.g. Latané, 1996).

A CA model consists of a grid whose cells each contain an automaton. This grid might be a rectangle or, if no edges are desired, a torus; the size and shape of the grid selected can influence the results substantially, for example encouraging clustering in a corner. Each automaton has a state: a race and a "racial tolerance" in the case of Schelling's model. It changes state in discrete time based on very simple, homogeneous rules concerning the states of neighbouring cells and its own previous state (Hegselmann and Flache, 1998). E.g. if 70% or more of my neighbours are of another race, I want to move.

This idea of local interactions producing a macro effect approaches our idea of complex systems. Hegselmann and Flache (1998) outline the main advantages of using CA: they model the essential aspects of many real life social processes, especially unintended macro consequences of individual actions and choices, such as Schelling's segregation or our class polarisation effect. They allow for quantitative explanations and predictions, as with traditional types of analysis, but crucially they also enable qualitative analysis. CA can even reveal flaws in a theory's underlying assumptions, by showing the unanticipated consequences through the micro-macro relationship.

Macy and Willer (2002) consider CA to be too rigid for many applications, as no two nodes can share the same group of contacts, yet there is no reason that could not occur in real life. Similarly, each node must have the same homogeneous pattern of relations, although it is possible to adapt CA to use irregular grids that overcome this constraint. Cederman (2005) notes that if, for example, a friendship network is being studied, then a static grid is too restrictive as well: it should be possible to dynamically alter the structure of the grid.

The primary criticism Gilbert (1995) makes of CA is that they model individual people as over-simplistic units. Whilst acknowledging that they can provide valuable insights into processes, he feels that it "requires a lot of theoretical imagination to move from patterns

of cells on a grid to conclusions about societies". A richer approach is found in multi-agent simulations, which endow the automata with cognitive capabilities.

2.5 Multi-agent simulations (MAS)

MAS have only been in use for a short time: they were developed in the '90s (Sawyer, 2003). They retain the bottom-up approach of treating the unit of observation as the individual, like CA and microsimulation, but model the individual (now termed 'agent') as a much more sophisticated entity. MAS evolved out of CA modelling, so earlier models tended to take place on a CA-like grid with local interactions only: they were essentially CA with more intelligent automata. Later models were more likely to break away from CA tradition, not necessarily even involving a grid at all (Gulyás, 2005). Object-oriented programming was another major influence on the development of MAS, providing the concept of self-contained objects with a natural correspondence to real-world entities. Artificial Intelligence provided the next essential 'ingredient' of endowing each object with intelligence, but instead of the focus being on each individual's intelligence, MAS is interested in the interactions between intelligent agents (Sawyer, 2003). An intermediate field, Distributed Artificial Intelligence, incorporated this shift towards interactions, but the final step towards MAS was to decentralise control and give each agent autonomy (ibid). Decentralised control is linked to the idea of self-organising complex systems. Autonomy means that the agent can refuse to perform tasks requested of it (or negotiate the conditions). This is the main characteristic that differentiates agents from objects, which must carry out any task asked of them.

2.5.1 What are MAS?

Not every multi-agent system is a simulation, but a survey of multi-agents systems as a whole, which cover applications as broad as industrial process control, combinatorial auctions and network routing (Sawyer, 2003) is out of the scope of this review. Other types of multi-agent system tend to have producing the system as a goal, whereas MAS wish to analyse the system. MAS themselves cover a large range of application domains, not just social simulation: economies, biological populations, computer or road traffic networks and games, for example (Luck et al., 2004).

Historically, there has been much disagreement on what exactly constitutes an 'agent'; each paper presents its own list of necessary characteristics. Davidsson (2000) captures the fundamental problem when he talks of a "continuum" of agents; what Gulyás (2005) says "ranges from ordinary object tokens to full-fledged AI-type agents". Bonabeau (2002) takes the extreme view of labelling even differential equations as agents because each describes "the dynamics of one of the system's constituent units", but this definition seems to cover so many things that it is no longer useful. For the purposes of this project, we shall define agents as *objects, situated in some environment they are able to sense, that exhibit adaptive, autonomous, goal-directed behaviour and can communicate with other agents through some predefined communication language*. They may or may not be mobile entities that are assigned a specific location in the environment, as long as they are situated somewhere in an environment, rather than being a 'disembodied intelligence' like an expert system (Jennings et al., 1998). 'Mobility' refers here to moving around in space within the simulated environment, rather than on a physical network like the Internet.

Goal-directed behaviour (often termed 'proactive') refers to agents taking the initiative to satisfy their design objectives. This could be a head teacher starting a marketing campaign, or a household deciding to move to another catchment area. To be effectively goal-directed, an agent must also be reactive and adaptive; it cannot rely on environmental conditions remaining static, allowing it to blindly follow a single plan

(Wooldridge, 2002, p.25). Wooldridge also warns that an agent should not be ***too*** reactive, or it will dither back and forth between plans as minor changes come and go in the environment, never accomplishing anything. This will not be as difficult a problem for our system as it is for ones with more continuous environments, as changes are expected to happen infrequently: for example, new data about school rankings will only be provided once a year. Finally, he notes that communication between agents should not simply take the form of exchanging data. There should be a social element to it, such as cooperation or negotiation. Our model will probably not extend very far in this direction, as the agents are not striving to complete tasks, so the need for interaction is less than in many MAS. The model may, however, consider social influence, which would certainly require social communication.

This definition puts us somewhere in the middle of the continuum of agents; while purely reactive objects would be too simplistic, we must avoid Gilbert and Doran's "trap of verisimilitude" (2005, p.12) whereby unnecessary sophistication is added to the model purely because it seems plausible, rather than being required. Most simulation researchers are in agreement that it is vital to give a model exactly the level of complexity required, and no more (e.g. Brenner and Werker, 2006**;** Macy and Willer, 2002) as superfluous detail (by definition) does not help, and may confuse. (Gulyás notes that a minority of researchers only consider simplicity necessary because better MAS tools have not yet been developed; whatever the reason, simplicity is seen as desirable at present.) It seems unnecessary to give our agents advanced AI planning capabilities; the lengthiest plan our model would consider is that a household might choose to relocate in order to change catchment area, but we will simply represent this as a choice, followed by a probability that the action is carried out successfully. However, it is also important not to over-simplify the agents, invalidating the modelled dynamics' realism, although Macy and Willer (2002) call on researchers to resist pressure to build realistic models if they would become as difficult to interpret as the target system itself.

In modelling these agents, a paradigm must be chosen for encoding their intelligence. The paradigm chosen should depend on the complexity of the agent. Purely reactive agents tend to use production rule systems with rules of the form 'if <condition> then <action>', which fire upon receiving the necessary environmental input (Gilbert and Terna, 2000). This may be the most common type of agent used in the social sciences (Cederman, 2005). At the opposite end of the continuum, 'cognitive' agents use more advanced forms of intelligence. They contain an internal representation of the environment, which they use to construct plans of how to achieve their goals (Sawyer, 2003). This is commonly done via the BDI (belief-desire-intention) architecture. Such complexity is not necessary for this project – since the desires of the agents are one-dimensional; we do not need to model cognition itself realistically; and long plans of action are not appropriate for a simple school choice – rather, it is the dynamic interactions of choices that need to be concentrated on. This could be represented adequately by a production system. Alternative possibilities for internal agent representations are considered in section 2.5.6.

2.5.2 Why use MAS?

MAS excels in models in which many heterogeneous agents interact, each perhaps behaving somewhat differently, creating dynamic, emergent system-wide effects. This makes it ideally suited to social sciences research (Davidsson, 2000). It suits the recently popular view that human societies are complex systems, non-linear and self-organising, much like flocks of birds, as it may be more useful to model such dynamics bottom-up through local interactions (Tesfatsion, 2003; Vicsek, 2002). Agents can adapt both at the individual and at the population level. The population level adaptation is evolutionary, through selection, social influence and so forth (Macy and Willer, 2002). MAS is also the

simulation technique in the best position to make use of the wealth of qualitative studies of social processes in the social sciences, namely when designing individual agent behaviour (Moss and Edmonds, 2005a).

Many MAS researchers are interested in the potential of MAS for investigating the two-way micro-macro link in emergence. It comes into play when simulating human societies, because unlike, for example, ants, humans can recognise and reason about institutions, and this "second-order emergence" may be crucial in understanding human societies (Gilbert and Troitzsch, 1999, p.11). For example, a human being's attitudes and beliefs might be affected by an emergent norm (Gilbert and Conte, 1995, p.11). Earlier forms of simulation were only able to investigate one-way emergence such as "vertical" emergence where, for example, microbehaviour generated macrophenomena. Similarly, "horizontal" emergence refers to spontaneous changes on the same level of scale in a model, what we might term co-evolution in the context of CAS. However, to study emergence comprehensively, "circular" emergence that cycles between micro and macro levels must be considered (Conte et al., 2001).

In economics, there exists the notion of the 'representative agent', which is a single entity that stands for every individual in the economy; the idea is that individual differences are averaged out in a large dataset anyway. This notion is used in the systems dynamics approach taken by Room and Britton and works well for forecasting aggregate data, but can only facilitate limited understanding of the underlying processes. Albin (1998, p.22) sees two possible routes to take: either introduce several heterogeneous agents, or create several homogeneous agents that are copies of the representative agent, but do not give them all the same data. Heterogeneity and bounded rationality (limiting agents' knowledge) are both central tenets of MAS.

Heterogeneity allows the distribution of behaviour to be analysed, rather than working immediately with the averaged data (Gilbert and Troitzsch, 1999). Equation-based models can take this aggregate data to great detail, for example assigning a distribution of values for a property to the representative agent (Sawyer, 2003), but it is nonetheless aggregate data that does not allow individual behaviour to be analysed. This makes methodologies like MAS that support heterogeneity especially valuable for social simulations; physical systems tend to have more homogeneous components (Gilbert, 2004).

The problem with perfect rationality, which is traditionally used in the social sciences, is two-fold, as first challenged by Herbert Simon. It is unrealistic to assume every agent will have ***all*** available knowledge, just as it is unrealistic to assume it has infinite time and resources at hand to find the mathematically optimal solution from the information it has (Albin, 1998, p.23).

Bounded rationality addresses these problems by limiting each agent's knowledge and processing capability so that the agent seeks only an approximately optimal solution. This is more realistic, at least in social domains, because people are often more interested in finding merely an acceptable solution rather than expending a lot of effort on finding the optimal one (Albin, 1998, p. 67; Bowe et al., 1994). If a person does try to be 'rational' and find the optimal solution, they are unlikely to truly possess all the pertinent information or necessary foresight (Axelrod, 1997**;** Fowler and Smirnov, 2005). While bounded rationality may sometimes underestimate the abilities of agents in the target system, it is likely to be a closer fit than perfect rationality, argues Makowsky (2006), although he adds that the real test lies in whether the model can be validated by empirical data. Validation is especially important for boundedly rational models because there are typically many possible ways of being less rational, but only one way of being perfectly rational (Gulyás, 2005).

Some researchers say that perfect rationality can cope with exponentially difficult computations, while bounded rationality can go no higher than computations of

polynomial complexity (Axtell, 1999). The justification for this seemingly arbitrary partitioning is, however, questionable – Board (1994) explains it is due to the common definition of tractable algorithms in computer science as being bounded by polynomial time, but does not examine how this applies to human brains. However, agent-based models are easily able to enforce this constraint, as they directly implement the decision-making algorithm (Gulyás, 2005).

Axelrod (1997) does not believe that anyone thinks that perfect rationality is realistic; he thinks that its main value lies in allowing agents to use deduction. He considers the main alternative to perfect rationality to be adaptive strategies, rather than aiming for optimisation. Adaptation in this sense could involve individual learning or take place at the macro level through 'survival of the fittest', like genetic algorithms (see section 2.5.6). The outcome of these interactive processes of adaptation is difficult to deduce formally, and so simulation is used.

Researchers often incorporate bounded rationality into their models by limiting agents' interactions locally, like CA do. There are many variations on this concept to make it more representative of real social interactions. For example, Mosler (2004) constructed a social influence simulation to examine the spread of environmentally responsible behaviour in a society; to incorporate realism, he limited interactions between agents to a set number of friends and a single stranger in each time step. In a perfectly rational model, the agents would have been fully connected and therefore able to consider the distribution of opinions throughout the population before deciding whether it was worth their while to conserve environmental resources. Mark (1998, cited in Macy and Willer, 2002) instead limited interactions by giving a higher probability to social interactions between agents with greater cultural similarity. Our model could simplify cultural similarity to be class-based, as it is likely that people of the same social class would mix outside their immediate pool of neighbours but nevertheless more often with people in the same circumstances. We could reduce the probability of interaction with distance as well, to take account of different catchment areas being dominated by different classes, so if a working-class family were in a middle-class area, their interactions would be more likely to cross class boundaries. Limiting agents' information like this causes each to develop a "subjective reality" as they assume that the sample of data made available to them is representative of the whole (Makowsky, 2006). This effect is especially suited to Makowsky's simulation of urban crime, but may well apply on a more general level to societies as well.

Another approach limits agents' knowledge more directly by placing a limit on their memory, varying it between agents to create heterogeneity as well (Tesfatsion, 2003). Fowler and Smirnov (2005) use a memory parameter in their simulation of elections to determine how new information is interpreted depending on past information on voter preferences and turnout.

Herbert Simon also suggested limiting computational accuracy, not just ability (Makowsky, 2006), but this does not seem appropriate for our model, in which the calculations performed by agents are not especially difficult, and so they would not be expected to make mistakes in real life.

Moss and Edmonds (2005a) identify a further advantage of MAS: it represents a compromise for sociologists who are critical of generalising from case studies, as varying the initial conditions of the simulation can generate a large amount of different data, and yet qualitative analysis is still possible, through examining the behaviour of individual agents.

Mosler (2004) admits that there is a large gap between a MAS and its target systems, but argues that there is no other alternative. If we want improve our understanding of fundamental social processes, we must build simplified, bottom-up models of them, and

MAS possesses the greatest realism out of all the techniques available to social scientists. Starting from a world that closely resembles the target world, MAS directly "grows" the simulated phenomenon, showing that we know how it was generated (Epstein and Axtell, 1996). It provides a "bridge" between the micro and the macro levels (Macy and Willer, 2002).

2.5.3 MAS vs Equation-Based Modelling (EBM)

The differences between MAS and EBM are of particular interest to this project, since we are proposing to improve upon an EBM with a MAS. Both support the exploration of system dynamics, but MAS allow the mechanics of the micro-macro relationship to be studied, as well as the trajectories taken towards equilibira, not just the equilibria themselves (Batten, 2000, p.21). MAS are the more intuitive tool: "many social scientists find [equations] either impenetrable or incredible as descriptions of social reality" (Tesfatsion, 2003).

Parunak et al. (1998) compare these two types of model. They argue that often, an agent-based model of policy implications can seem more natural, and even a simple MAS can exhibit surprisingly illuminating behaviour. They consider the main differences between agents and equations to be two-fold.

Firstly, a set of equations is focused on modelling relationships between observables, while an agent-based model explicitly models behaviour, so that any relationships between observables that emerge are the output of the model, not its input. This means that if we can build agents with behaviours that generate the same end effect as Room and Britton's model, we can examine how the dynamics of the interacting behaviours work to produce the cumulative effect, such as a trajectory towards equilibrium. It is easier to run direct 'what-if' experiments by manipulating the agents' behaviour directly, and consequently easier to draw policy implications from the results of the experiments, as the causal behaviour will be known. (A drawback of this is that the modelling of behaviour is more complex, and therefore more error-prone.)

Secondly, equations tend to work with system-level observables rather than individual attributes. A multi-agent model would allow for boundedly rational heterogeneous agents with different attributes, strategies, and information, which is not possible in a set of mathematical equations (Axtell, 1999). Although it could be argued that individual differences are effectively averaged out when the outcome is analysed, it would be interesting to be able to examine the behaviour of individual agents in order to gain a deeper understanding of the process behind the social exclusion effect.

Parunak et al. also point out that MAS are capable of expressing much more complex situations than EBM. This could be said to be a question of scalability: it is easy to inject another agent into a simulation – no new code need be written – but adding another variable to an equation increases the analytic tractability, and decreases transparency. The same principle applies to adding new behaviours (Gilbert and Conte, 1995). Room and Britton had to constrain their model to only consider two schools because of this, and so one use of a MAS implementation would be to examine the consequences of involving additional schools. Having multiple schools could allow us to consider multiple catchment areas, and would allow the option parents hold of moving house to another catchment area, currently included as one of the contributing factors to middle-class parents' higher preference for the middle-class oriented schools, to be included explicitly as a behaviour that middle-class parents are more likely to carry out. Not being confined to easily manipulated symbols widens the options considerably. Axtell (1999) adds that physical or social networks, such as might be needed to represent catchment areas and social influence in our model, are difficult to incorporate into an EBM.

On the other hand, the greater elaboration of MAS means that drawing conclusions is more technically demanding, as rigorous statistical tests must be applied (Gilbert and Doran, 1994, p.3). Equations also have the advantage of allowing provable mathematical conclusions to be derived about, for example, the stability of the equilibria (Room and Britton, 2006a**)**. Gilbert also notes that since MAS typically have a larger number of parameters and underlying assumptions than EBM, difficulties can be encountered in finding empirical data to justify them, as well as in the computational load of performing a sweep of the entire parameter space (see section 2.6). Of course, once these assumptions have been justified, the theory may be better understood as a result. A related risk is that verification and validation can be a lengthy process, as they must proceed experimentally rather than constructing a mathematical proof, as for an EBM.

Gilbert and Troitzsch (1999) address some other practical issues. MAS can be designed in a modular fashion, so that differing theories can easily be compared by changing one part of the model without needing to modify the others. These subsystems also provide an intuitive way of dealing with concurrent processes. Bonabeau (2002) adds that MAS enable a more sophisticated form of stochasticity, while an equation must content itself with the addition of a "noise term".

Edwards et al. (2003) compare a MAS and an EBM implementation of a particular theory, concluding that the greater power of MAS is not always necessary. If the global behaviour is the same as with the aggregate EBM, then the shorter computation times, and the useful insights offered merely by examining the structure of an equation before it has even been run on data, indicate that EBM should be used. This is arguable (although it is true that the value of EBM should not be minimised) because the understanding gained from seeing how micro-behaviour generates macro effects can be valuable. MAS is "the canonical approach to modeling emergent phenomena" (Bonabeau, 2002). One could also say that it is actually desirable for a MAS to generate the same macro effects as an EBM because that is a form of model-to-model validation (see section 2.6).

Batten (2000, p.249) puts forward that "economic development depends critically on path-dependent principles of self-organisation and coevolution, unfamiliar processes that have remained largely untouched by traditional analytical tools". This is precisely the motivation for re-implementing Room and Britton's EBM as a MAS.

2.5.4 Issues in using MAS

One issue to address is how to represent the target environment in the simulation. It does not necessarily need to be spatial, giving each agent a location, although in our case we will need, at a minimum, a crude spatial division according to catchment area. The environment will need to be given an extent to accomplish this, specifying its size, boundaries and perhaps its shape (Odell et al., 2003). Each square in the grid could be related to others through a direction (e.g. 'diagonal') rather than an absolute address; this kind of relationship is useful if agent interactions are based on proximity (Odell et al., 2003). An alternative to a spatial network of agents is a social network that specifies possible interaction partners independent of location (Gilbert, 2004). This network may have relational rather than absolute links, so that one agent is related to another agent, which in turn is related to several others – these links could be followed for a set depth. Our model could use both a physical and a social network if social influence or other relationship-dependent forms of interaction are to be modelled.

In a deterministic environment, the next state of the environment is uniquely determined by the current state and the actions selected by the agents (Odell et al., 2003). This model's environment will be non-deterministic because of the stochastic elements it will contain. But even if the environment were deterministic, it would still not be predictable or controllable by any one agent, because the agents have no control over, or

foreknowledge of each other's actions (including whether their own actions will have the desired effect).

The environment is often modelled as another object with attributes like time and other global data, and possibly a buffer for messages (e.g. Gilbert, 2004**;** Odell et al., 2003) although if a MAS framework were used, it would handle message-passing. However it is represented, it should be modelled explicitly, even though agents are not external to it: they are "embodied" in it (Helleboogh et al, 2007). Forcing agents to communicate via an abstraction like the environment (for example, by requesting a list of neighbours, or by posting messages on a shared noticeboard) rather than directly simplifies the design (Gilbert, 2004). The *simulated target environment* should always be decoupled from the *simulation environment* (i.e. the infrastructure running the simulation) as another principle of good design (Helleboogh et al, 2007).

An interesting question is whether to have a single centralised environment, or an environment for each region (here, catchment area) that oversees all the environmental processing that takes place within its bounds (Odell et al., 2003). The centralised environmental design is probably more suited to our model because there will be significant communication and movement between boundaries.

As well as the representation of the environment, the representation of emergence should be considered. Conte et al. (2001) present a symposium where a number of modellers argue their case. Moss' position is that only dynamic system properties should be modelled indirectly through emergence; static properties should be explicitly built in. He also argues that if the purpose of the model is to show interaction with norms or other global properties, as opposed to showing how such properties are generated in the first place, then a top-down approach of specifying these global properties explicitly is most suitable. Sawyer argues more strongly that the top-down approach is often necessary, not just most appropriate, by drawing on sociological theory which suggests that the assumption that macrosocial entities do not truly exist in their own right has limited explanatory power. Regardless of whether they do exist, it can be useful to 'pretend' that they do. Edmonds disagrees, saying that it is "more profitable" to take a bottom-up approach when first building understanding, and that top-down models can then follow once it is clearer what they should be modelling. This is what Sawyer denounces elsewhere (2003) as not modelling true emergence but rather only partial, horizontal emergence. Conte prefers an approach that mirrors real life, where an emergent phenomenon first appears naturally, and must then be modelled explicitly in order to investigate the two-way interactions between it and the micro-level. Sawyer (2003) later proposes the same idea, explaining that the model would need to be capable of dynamically restructuring itself at run-time in order to represent the newly emerged macro phenomena. Such a simulation platform has not yet been developed, and could pose significant difficulties. Modelling second-order emergence convincingly is generally held to be an unsolved problem in MAS (e.g. Sawyer, 2003; Gilbert, 1995).

Less controversial is the idea that macro phenomena can affect individual agents even if they have no internal representation of them, as long as they are context-aware (Sawyer, 2003). However, some researchers (e.g. Cederman, 2005) do disagree, and see this as an outstanding issue in the MAS community: that richer cognitive models need to be developed for agents.

2.5.5 Event scheduling

MAS shares similarities with discrete event simulation (DES). In DES, a list of scheduled events advance the state of the system, depending on its previous state. An event may modify the event list itself. A DES may be time-driven in discrete steps, or alternatively, if the simulation is not to be monitored in real-time by humans, it can be event-driven,

which effectively 'fast-forwards' to the next scheduled event on the basis that we know nothing will happen in between (Davidsson, 2000). The hybrid approach combines continuous and discrete phases (Helleboogh et al, 2007). It seems that a school admissions simulation could benefit from a discrete or hybrid approach, as it would be inefficient to simulate every time step when most – perhaps all – of the activity will be concentrated around the time when applications for places are submitted. Hegselmann and Flache (1998) warn that discretisation can produce modelling artefacts if the target environment is continuous, but this does not seem to be a danger for our model.

Davidsson expresses scepticism that MAS is particularly suited to event-based simulation, arguing that there would need to be a central coordinator to keep track of events, which he sees as being contrary to the design principles of MAS; or highly synchronised agents, which would impose performance penalties. However, it seems that MAS toolkits may provide an acceptable solution, as they provide advanced capabilities like parallel discrete event scheduling. In another application domain, with more highly autonomous agents, Davidsson's argument would appear more relevant, but in this case, the idea that parents must submit applications for school places at a certain time of year, and that head teachers will then process these applications upon receipt, will already require a degree of central coordination. There is a direct parallel to this in real life, as the Council might send all parents a letter reminding them to apply for a school place. However, it could be that having designed an initial prototype simulation for this project, Davidsson's view will need to be revisited if unanticipated problems are encountered with event-based simulation.

While it may be useful to borrow the concept of event-based scheduling from DES, the MAS approach still has the advantage over DES for our purposes. It is much more scalable than DES, as new agents can easily be added, even dynamically (Davidsson, 2000). Additionally, a DES modeller is forced to place restrictions on the choices entities can make in order to fit them smoothly into the DES framework, although this can be overcome by 'hacks'. It would, therefore, not support the simulation of co-evolving strategies well. More generally, it also does not use the agent metaphor that facilitates communication with non-programmers. Dubiel and Tsimhoni (2005) show how these limitations can be overcome (to a certain degree) by incorporating an agent module into a DES. However, their main motivation for this seems to be to allow them to build on existing DES models, so it does not provide this project with any reason to pick DES or a merging of the two over MAS.

2.5.6 Other adaptive simulation techniques

Evolutionary algorithms, such as genetic algorithms (GA), are a natural technique to consider for modelling co-evolutionary processes. GA assigns a 'degree of fitness' to each individual in a society according to some metric. Each individual is represented by a string of code that forms a strategy, according to some encoding scheme of possible behaviours (Macy and Willer, 2002). In Room and Britton's model, the parents whose children attended the highest-rated schools, and those schools' head teachers, would be the 'fittest' individuals, so they would be 'interbred' with others of their kind to produce new combinations of strategy that might improve on the original ones. In addition, random 'copying errors' are introduced to provide heterogeneity. Over several generations, the overall fitness of the population would increase as it adapted to the new socio-political environment (Gilbert and Terna, 2000). This co-evolution of parental strategies on the one hand, and head teachers' on the other, could be expected to lead to the same class polarisation indicated by Room and Britton's model.

GA are typically used to search for either optimal or likely solutions. If we were interested in finding a social policy to optimise social welfare, a global approach could be

used where each strategy is aware of every other strategy, and 'mates' with them with equal probability. This project is instead aimed at understanding the likely (rather than optimal) effects of the situation, so it would limit each string's knowledge and interactions locally by placing them in a spatial network, like CA (Macy and Willer, 2002).

Another type of adaptive simulation is a neural network, which was used by a recent social mobility and inequality simulation (Meraviglia, 1996). This technique, like GA, is well suited if strategies need to adapt to changing circumstances.

Neural network simulations use the metaphor of connections between neurons in the brain. They contain many simple neurons that are linked by weighted connections, and work together towards the simulation goal in interconnected layers that process the input according to their weighting, and then pass it on (Gilbert and Terna, 2000). The neurons are capable of adaptive behaviour by means of changing these weightings through the application of a "learning rule" (Meraviglia, 1996). This may, for example, take the form of "training" the network towards some pre-defined outcome, if the exact process leading to an outcome – such as class polarisation – is unknown. However, the process identified by the neural network cannot be expected to correspond to the real-world causes.

While Meraviglia's findings agree with those obtained through more traditional techniques being applied to the data, showing that neural network simulations can work for sociological questions, breaking down the school system to networked variables does not seem like a very intuitive method to further our understanding of the underlying processes. It seems that a MAS, with self-contained agents following rules understandable on a high level, is a better match for our purposes. Gilbert and Terna (2000) also point out that for both GA and neural networks, a particular level of interest has to be set at which to work, for example individual children, households, regions and so on, as a single neural network could represent any one of those. If an entire society is modelled, then it becomes difficult to give each individual in the society the sophistication that an agent in MAS can have.

Instead, a MAS could provide the scalable agents infrastructure, but not fix the agents' decision module implementation. GA and neural network libraries are available for MAS that could be 'plugged into' a decision module. This would still not provide a good high-level view of the strategy being followed like traditional production rules would, but it might nevertheless be an elegant solution. However, Moss and Edmonds (2005a) point out that not having an intuitive correspondence from the agent implementation to the real life equivalents they represent brings a more serious problem with it than losing the benefit of intuitive understanding: it also makes it very difficult to validate the model qualitatively, as individual behaviour cannot be analysed. Since our model is geared towards furthering our understanding of the underlying processes, not just prediction, qualitative analysis will be an essential tool towards that end, and so we must look for alternatives to GA and neural networks for the agents in this model.

2.6 Validation and verification of MAS

Validation and verification are important for any software product, but they are especially crucial for simulations in order for their output to be regarded with any seriousness, and it is in their output that their usefulness lies (Gulyás, 2005). This is because of the inherent difficulty of distinguishing unexpected results from software faults, so the effects of a bug could easily be misinterpreted as an emergent phenomenon (David et al., 2002).

Moss and Edmonds (2005b) allege that "in the social sciences there is typically little or no attempt to validate theory". If this is true, then one purpose of simulation could simply be to validate a theory, although Moss and Edmonds point out that if a simulation relies

upon a previously unvalidated social theory, it cannot be viewed as validated itself – a difficult condition to satisfy if their earlier allegation is correct.

In the cross-disciplinary field of social simulation, it is important to define what 'verification' and 'validation' mean, as computer science and the social sciences attribute different meanings to the terms (David et al., 2005). Program verification in the social sciences often takes the form of persuasive verbal argumentation (ibid), but we would like something more rigorous to vouch for the correctness of our simulation according to its specification. At the same time, validation of a social simulation must incorporate the interpretative approach of the social sciences to judge how well the simulation represents reality (ibid).

David et al. (2002) divide validation and verification of simulations into static and dynamic activities, while most other modellers, e.g. Balci (2003) and Bryson et al. (2006), consider only a subset of them, as they do not distinguish between checking the simulation against the conceptual model and against the target system.

1. *Static validation* is concerned with validating the underlying theories and assumptions of the model; Graham Room's research and that of other sociologists covers this. We will simply take note that if the theory is wrong, then our model may simply confirm the error (Gilbert and Doran, 1994, p.4), and that the possibility that another theory could explain the results equally well, or even better, can never be fully eliminated.
2. *Static verification* entails checking that the simulation is an accurate representation of the conceptual model (which in our case corresponds to the model described by Room and Britton); this is akin to checking that an ordinary computer program meets its specification in standard software engineering, and to what Balci calls "transformational accuracy".
3. *Dynamic verification* checks the simulation outputs against those of the conceptual model.
4. Finally, *dynamic validation* aims to determine that the simulation outputs are correct according to the target system, like Balci's test of "behavioural or representational accuracy".

Both the dynamic techniques correspond to testing in software engineering. David et al. note that successful validation hinges on successful verification, i.e. if there are severe enough errors in the code, the validity of the model must be questioned even if validation tests succeed.

Like David et al., Küppers and Lenhard (2005) think that the three levels involved in simulation – the target system, the conceptual model and the simulation itself – have an impact on the way simulations should be validated. They identify a common view that since simulations are equivalent to theories, they should be validated the same way theories are in the social sciences. They disagree, saying that simulations, rather than being models, are "models of models" that mediate between theory and reality, and as such should be treated differently: as an attempt at imitation of the underlying dynamics of the conceptual model. This attempt should be carried out with a "quasi-empirical" evolutionary programming approach of refining the model to better fit empirical data: since simulations are not direct numerical ***solutions*** of theories, they cannot be validated by showing that they can be derived from theory. Rather, empirical data must be drawn upon.

Unfortunately, judging the goodness of fit against empirical data is complicated if the model contains stochastic elements, although methods of statistical analysis can overcome this (Karlsson, 1969). Comparing simulation results with empirical data also requires a greater degree of rigour when specifying parameters: rather than only the signs

and relative magnitudes of parameter values being important, when "calibrating" the model to fit empirical data, the parameter values must be tuned exactly to correspond with historical data (Makowsky, 2006). Finding such exact parameter values may be too demanding a task for this project; they would perhaps require specific field data to be gathered.

Zeigler (1975, cited in Küppers and Lenhard, 2005) identifies a stronger form of validity: structural validity, which applies when a model's basic interactions mirror the way in which the target system generates the behaviour. Clearly, MAS is currently the most suitable method for achieving structural validity, as agents can be designed to make decisions analogously to the way human beings do, but even so a MAS will require simplifications to be made to avoid building a system that is just as complicated as the target system. Küppers and Lenhard (2005) acknowledge that social simulations cannot achieve Zeigler's strict definition of validation.

Dynamic validation and verification are closely related to actually running the simulation to gain data for analysis. In both cases, a "parameter sweep" must be undertaken, examining the effects of varying each parameter over their ranges, including the interaction effects between parameters; in the case of validation, this will be to conduct a sensitivity analysis. Conducting a comprehensive parameter sweep requires non-viably high computational resources if the simulation is non-trivial, especially if non-linear parameter relationships are involved, as these are difficult to detect if parameters are only modified in isolation (Miller, 1998). This is further exacerbated by stochastic models, as the random seed should be varied for each parameter set as well, for example using the Monte Carlo technique. A more tractable approach must either manually identify which parameter regions are "interesting", and which are most likely to "break" the model, which risks missing out important areas (although well designed hypotheses and null hypotheses should help with this); or use another program to identify the regions of interest automatically, according to some fitness function. Brueckner and Parunak (2003) took the second route with an adaptive multi-agent program designed just for this purpose. Similarly, Miller (1998) constructed a search algorithm to find parameter combinations that "break" a model's implications by trying to maximise the deviation from the expected results. Unfortunately, such advanced techniques are beyond this project.

A model can never be guaranteed to be free of bugs, and so the possibility that the results of the simulation are merely artefacts of bugs must always be considered. Rather than saying a model is "validated", one should say it is "better-validated" the more tests it passes (Bryson et al., 2006). MAS may be more susceptible to undetected bugs than other computer programs because of their distributed, self-organising nature, and the fact that a small anomaly in an individual's action may not have a significant impact on the macro phenomenon (Gilbert and Terna, 2000; Axtell, 1999). Herein lies the value of a sensitivity analysis: to highlight anomalous behaviour arising when non-critical parameters are varied. However, Gilbert and Terna stress that models of complex systems may be inherently sensitive to initial conditions, and so high sensitivity might actually be desirable.

Another validation technique, model-to-model analysis, has recently gained strong support in social simulations (Küppers and Lenhard, 2005). Edmonds and Hales (2003) report that model alignment is "very difficult, but very revealing". They found it exposed a number of bugs when they re-implemented an arbitrarily selected model that had already been published, and was generally accepted as valid, and expect that the same would apply to most published models (this is backed up by Axelrod (1997) who replicated eight different models). These bugs might not change the overall "statistical signature" of the simulation results, but the original verification process could still be said

to be lacking. Edmonds and Hales do not consider this surprising because they see verification as being of limited use when emergent phenomena are being studied, as it is questionable to specify them in advance as requirements that can be verified. David et al. (2002) argue that such emergent phenomena must be identified in retrospect and then verified anyway, it seems referring only to their concept of dynamic verification (defined above), which does not rely on a specification as static verification does, instead checking only outputs. Edmonds and Hales also found that model-to-model analysis had the ability to expose "hidden parameters" that had been unwittingly missed out from one of the models, when output was compared. Merely attempting to re-implement the model also revealed that the original model specification was ambiguous.

Our model has the advantage that there is already a model in existence that could be "aligned" with it using model-to-model analysis, to increase confidence in its validity. Equally, Room and Britton's model could enjoy greater validity if it agrees with another implementation of the theory, although it could be argued that their model is merely acting as the specification for our model, and so of course the results will agree. However, Hales et al. (2003) report unanimous agreement in a recent MAS workshop that aligning top-down models like EBM with bottom-up models like MAS is useful, with the cases where the models do not agree being of especial interest.

Edmonds and Hales recommend that a re-implementation of a simulation should use different types of programming language, and preferably be programmed by different people, to avoid simply repeating unfounded assumptions and mistakes. We go one step further and use an entirely different methodology as well, although the fact that one model (in Ostrom's terminology) uses an entirely different symbol system may make it more difficult to align them. However, it may be that due to limitations of the programming environment chosen for the initial prototype, a different framework must be chosen for a second implementation, in which case these issues will become more relevant.

Balci (2003) also suggests that a simulation should be certified by an independent expert to increase confidence in its credibility. While not independent, Graham Room, who has undertaken joint supervision of this project, may be able to fulfil this role to an extent, although, as with all stakeholders in a project, confirmation bias is a risk. Mosler (2000) considers such expert opinions to be especially important if sufficient empirical data is not available to validate against.

MAS offers an additional level of validation over more traditional simulation techniques, and also over more advanced techniques that do not try to imitate real individual behaviour, such as neural networks. Every type of model can be validated at the system level, by comparing the output against the expected output (perhaps empirical data), but agent-based models also allow local observations of agent behaviour to be validated, perhaps against qualitative accounts of target behaviour, preferably by domain experts (Parunak et al., 1998). Moss and Edmonds (2005a) call these techniques "macrovalidation" and "microvalidation", respectively, and declare a model that has been validated in both ways "cross-validated". A cross-validated model is more likely to display genuine emergence, rather than the 'emergent' properties being somehow built in, although Moss and Edmonds caution that the individual behaviour still only provides one possible explanation for the macro effect. This is of course a problem common to all scientific explanations, not just simulations.

Gilbert (2004) differentiates between qualitative macrovalidation, in which the general pattern of data is in agreement with the expected results, and the stronger measure of quantitative macrovalidation which requires the data to be in agreement according to tests of statistical significance. The latter measure will not be feasible if it is not possible to

obtain detailed empirical data for comparison. He also suggests running cross-sectional and longitudinal analyses as part of microvalidation.

2.7 Related simulations

School choice simulations have been run for a long time, although none that are multi-agent based, that the author is aware of. Hoyle and Robinson (2003) constructed a mathematical simulation in which they investigate the effects of school league tables on schools' performance and social class composition, and unlike us, the accuracy of the league tables in reflecting school performance. They do not, however, examine the two-way 'competitive assortative mating' effect where schools cherry-pick students: they only consider parental choice. Their model is more complex than Room and Britton's, in that it actually constructs league tables from pupils' exam results, but it is nevertheless fundamentally a mathematical model, and so does not bring the insights into the co-evolutionary process that it is hoped our MAS model will.

Manski (1992) constructed another mathematical simulation of school choice that is closer to our aims in nature, as it views schools as competitive firms and uses the market metaphor. Interestingly, Manski considers bounded rationality: he is concerned that "ill-educated" parents might lack the ability to properly interpret information on school choices. Many sociologists share his concern (e.g. Lauder and Hughes, 1999, p.43; Karsten et al., 2001). However, the restrictions of equations do not enable him to model the issue, as we hope to using MAS.

An even earlier mathematical simulation by Birdsall and Meesook (1986) on Brazilian educational and income inequalities similarly regrets the low degree of heterogeneity it is able to express among families. Yair (1996) takes a "network analysis" approach to the Israeli education market, which appears to be a form of microsimulation, but his study too suffers from the limitations of the methodology. The need for a school choice simulation that addresses these issues using a more powerful technique is apparent.

Fowler and Smirnov (2005) present an example of how this might be done with their MAS of political parties and voters, analogous to our schools and parents. The political parties even adopt niches based on the current climate, as schools can (Plank and Sykes, 1999). They limit the voters' information about population preferences to that of their local neighbours', embedding them in a social network. Social network considerations may be highly relevant in the domain of school choice as well (Lauder and Hughes, 1999, p.44).

2.8 MAS architectures and toolkits

Having established that MAS is the methodology most suited to extending Room and Britton's work, the question of what supporting framework to use remains. There are a wide range of MAS toolkits and libraries freely available, although many are still in the early stages of development (Luck et al., 2004), and so commercial products will not be considered. Since there are such a large number of platforms that support simulation in particular, more general multi-agent frameworks such as JADE will not be explored.

There are some fundamental requirements that the agent architecture must meet. It must provide an environment into which the agents can be injected, and given a physical location, as well as supporting more abstract social networks. It must also provide a communication infrastructure, and scheduling of events. It would also be useful for the analysis if the framework automated functions such as running multiple simulations with varied parameters; inserting "probes" to examine the internal states of agents, and the dynamics of agent interactions; as well as various logging and statistical analysis

techniques, but such analysis can be supplemented by "add-on" programs if necessary (Brueckner and Parunak, 2003).

Swarm[1] is the best-established agent simulation tool, and has the advantage that a Swarm simulation is more easily replicable by other researchers, who will be familiar with the development paradigm (Gilbert and Terna, 2000). It supports MAS written in either Objective C or Java; these are both object-oriented programming languages, which are the natural choice for a MAS due to the close correspondence of agents with objects. MAML is a high-level macro-language for Swarm designed for users without strong programming skills (Gulyás, 2005). However, Swarm is a general-purpose artificial intelligence simulation package (Tobias and Hofmann, 2004), and since Repast and MASON (below) are essentially based on Swarm but oriented towards the social sciences, they seem more suitable than Swarm, and therefore MAML, for our purposes.

Repast[2] appears to be the most suitable candidate for an initial prototype, as it is specialised towards our application area, relatively mature with a well-developed user base, and is easy to use (Tobias and Hofmann, 2004). Several researchers view it as "the most suitable package for modelling complex social systems" (Gulyás, 2005, p.155). If, having evaluated the initial prototype, it appears that Repast does not provide sufficient flexibility for further refinement of the model, a more open and extendible framework such as AgentScape[3] could be considered for a second iteration. AgentScape is not being used initially because it does not provide any of the built-in data graphing and logging tools that Repast does; its immaturity also poses a greater risk with lack of documentation and probable bugs. Nor is it specific to social simulations. Both Repast and AgentScape require agents to be implemented in Java, again an object-oriented programming language.

The agent's decision-making ability could be implemented using a logic programming add-on to Java such as JBoss Rules[4] (formerly known as 'Drools'). This production system would allow for a more natural representation of their reasoning.

Another toolkit that was considered is MASON[5], which is modelled after Repast (Tobias and Hofmann, 2004), or the extension BOD MASON[6], but they are geared towards reactive artificial intelligence and visualisation of mobile agents, and so Repast seems a better choice. Experience at this University with Repast on similar projects makes it a favoured candidate as well. SymBioSys[7] (McFadzean and Tesfatsion, 1999) is a C++ library for evolutionary simulations, but supports genetic concepts too explicitly for our purposes; we prefer to use them as a metaphor for more general behaviour.

Many researchers advocate using a new software development model, "agent-oriented software engineering" when developing multi-agent systems (e.g. Zambonelli and Omicini, 2004; Wooldridge et al., 2000). Since we are not creating cognitive planning agents with complicated communication protocols and so forth, it seems that a more standard evolutionary programming approach will suffice for this project.

[1] Available from http://www.swarm.org (accessed 23 April 2007)
[2] Available from http://repast.sourceforge.net/index.html (accessed 23 April 2007)
[3] Available from http://www.iids.org/research/aos (accessed 23 April 2007)
[4] Available from http://www.jboss.com/products/rules (accessed 23 April 2007)
[5] Available from http://cs.gmu.edu/~eclab/projects/mason/ (accessed 23 April 2007)
[6] Available from http://www.cs.bath.ac.uk/~jjb/web/BOD/BOD-MASON.html (accessed 23 April 2007)
[7] Available from http://www.kumo.com/~david/SimBioSys/ (accessed 23 April 2007)

2.9 Conclusion

We have seen that there is significant interest from sociologists in the mechanisms and dynamics of social exclusion that come into play during school choice, but that existing mathematical models are unable to provide them with sufficient insight into these dynamics, as well as issues such as population heterogeneity and bounded rationality. Given that the market-based school system can be classified as a complex system with co-evolving, self-organising strategies, it is time to harness the recently discovered power of MAS and analyse these processes from the bottom up, rather than contenting ourselves with a "black box" whose inputs and outputs we can examine.

The addition of this new level of elaboration will make it harder to analyse the data, to justify the new assumptions that will need to be made, and to validate the model to any degree of confidence. It is hoped that even if the model cannot achieve the rigour of validity required for successful prediction, that it will nevertheless aid our understanding of how the strategies of parents and schools interact to produce the class polarisation effect that is characteristic of the Baker reforms.

3 Software development

3.1 Requirements

Due to the exploratory nature of the project, precise requirements could not be specified in advance. However, the following key requirements describe some constraints upon the envisioned simulation. Use of the term "shall" indicates a mandatory requirement, while requirements denoted by "may" are desirable but optional.

3.1.1 Required functionality

1. A multi-agent simulation model shall be developed which is functionally equivalent to the equation-based model described by Room and Britton (2006b).
2. The simulation model shall use two main types of actor: head teachers, who pick a particular allocation policy; and middle-class or working-class parents, who choose which school to send their child to.
3. The agents shall be heterogeneous; that is to say, different head teacher agents may choose a different allocation strategy, and parents differ in the strength of their preferences, and in their reasoning.
4. The model may introduce extensions such as new parameters and new options, in order to evaluate their possible effect. It should be designed with extendibility in mind. Possible extensions include:
    a. Considering more than two schools, perhaps in different catchment areas
    b. Allowing parents to move from one catchment area to another
    c. Explicit representation of 'aggregate parameters' whose effect was previously only cumulatively considered, such as parental investment and teacher morale
    d. Investigating the effect of siblings already attending a candidate school
    e. Influence of a social network of neighbours on school choice, for example if neighbours wish to car-share
5. The model shall encourage experimentation through straightforward manipulation of parameters.
6. Analysis of the model shall consider explanations for the system-level patterns that emerge from agent behaviour.
7. The model may also be used for predictive purposes, depending on time constraints.
8. The model shall be verified and validated using appropriate simulation and agent-based programming techniques. Its predictive capabilities may also be measured against real life data, depending on availability.

3.1.2 Configuration requirements

1. The simulation shall initially be developed in an environment that encourages rapid prototyping. The second phase of the simulation may necessitate switching to a more flexible, open framework.
2. The development tools chosen shall be compatible with Eclipse if possible, assuming the use of Java as the primary development language. Since Eclipse is the dominant Java IDE (Geer, 2005), its use should make it simpler to change tools during the

development process if necessary, as we cannot establish in advance which tools will be needed.

3. Version control software shall be used, to facilitate the experimental development of extensions to the project

3.1.3 Functionality that is outside the scope of this study

1. The agents shall employ some form of learning so that the model is not too simplistic, but this may be closer to the concept of, for example, utility functions than complex machine learning models, as adding such functionality is too large a task for a project of this scale.

3.2 Design

The project was divided into two phases: first, building a model equivalent to Room and Britton's; and second, refining and exploring the model by incorporating new parameters and decision processes. The initial model may be built using a different technology platform than the second, as explained in the configuration requirements (section 3.1.2).

3.2.1 Initial prototype

The main purpose of the initial prototype was as a proof of concept, and for gaining some experience with agent-based simulation, so that more suitable technology could be chosen for the more refined model if necessary. It was therefore developed in a rapid prototyping environment, using incremental, evolutionary techniques. An agent toolkit with a large library of common functions, and a simple, easy to use framework for running simulations, was essential for rapid prototyping. The literature review already identified the social simulation framework Repast as a suitable candidate for this task (see section 2.8).

No production rule system was used at this stage, because a simple procedural translation of Room and Britton's equations seemed more straightforward. The agents' decision procedures could be rewritten as rules once it had been established that the basic concept worked as an agent-based model.

Adapting the macro equations for agent-level decisions

The original equations consider only two types of entity: parents and schools. Middle-class parents' preferences are described collectively by a function m giving the fraction m(x) of middle-class parents who prefer to send their child to the school where the fraction of middle-class students is x, as opposed to sending it to the other school[8]:

$$m(x) = \frac{1}{2} + a(x - \theta)$$

Equation 3.1 - fraction of middle-class parents choosing school with middle-class fraction x

where θ stands for the fraction of middle-class children in the population as a whole, and is assumed constant. Working-class parents' choices are represented similarly by the function l:

$$l(x) = \frac{1}{2} + b(x - \theta)$$

Equation 3.2 - fraction of working-class parents choosing school with middle-class fraction x

[8] Where the fraction of middle class children is θ – x.

where a and b are the strength of preference coefficients such that a, b > 0. It is also assumed that a > b, since middle-class parents are likely to favour a middle-class dominated school more strongly than working-class parents (Room and Britton, 2006b).

Thus, equations are used to describe population-level parental choices.

Only a single equation is necessary to describe the system-level behaviour of how many children of each class are allocated to each school. If a school has too many applicants, it must choose between them; too few, and it is forced to take them all, as well as the children rejected from its rival school. The system-level equation incorporates both the parental choices described above, and the schools' allocation policies, describing what fraction of middle-class children end up in each school as a result. Both schools are assumed to share the same policy: either *class blind*, where they choose students irrespective of social class; or *class sensitive*, where each school accepts as many middle-class children as possible, accepting working-class children only when no more middle-class children are left. The interested reader can find the formula in Room and Britton (2006b).

However, this model could not use the system-level equation, because it was intended to generate that outcome bottom-up out of choices made by individual schools and parents. It could, however, use the parental choice equations, and the informally stated 'allocate places at random' vs. 'accept all middle-class children first' school strategies. Its decision procedures are therefore much simpler than the equations presented in the original model, since they need only deal with local choices made by a single entity.

The parental choice formulae could not be used in their original macro-level form, of course: they required adaptation. Room and Britton's equations give the fraction of parents making a choice, but we wished to translate this summary of events into a decision procedure that each individual parent agent could apply. The simplest way of doing this was to turn the concept of a fraction of parents making a certain choice into the probability that each individual parent will make that choice. Approximately the same result should be obtained, albeit with added 'random noise' stemming from the particular sequence of random numbers used.

Schools were assumed to contain six year groups, so that the fraction x of middle-class children in a school is effectively the mean middle-class intake over the past six years.

Actors

Instead of using parent and school actors, the simulation used children and schools. This is essentially equivalent, but has a nicer semantic feel to it when a school is considering a child's attributes on application, rather than examining a 'parent' object for suitability. An alternative would have been to include both parent and child objects, forming a family unit, but the details of *how* a decision is arrived at, such as whether one parent has more say than the other, or if the child has any influence, is only of tangential concern to us. Rather, we are primarily interested in *what* decision they arrive at, and *why*. It is therefore more appropriate to abstract the family unit into the single concept of a 'child'.

Schools have the attribute *league table rating* instead of exposing a school's fraction of middle-class students directly, as Room and Britton's model does. The design allows for future expansion, but at present it is merely an average of the school's fraction of middle-class students over the past four years, since school league tables in the UK tend to include recent historic data as well. In reality, school league tables are of course based on exam results, but the school's social make-up is considered to be a rough approximation of this, since middle-class children tend to perform better in exams; additionally, their peers' results are positively influenced by a high percentage of middle-class children in their class (Lauder and Hughes, 1999).

The only attribute of interest held by children in the initial prototype is their social class. This has only two possible values: working-class or middle-class. A finer-grained model of social classes such as the eight-, five- or three-class versions suggested by the Office for National Statistics (2004) was considered, but not implemented, because other extensions seemed of greater potential interest. It could also have been challenging to correctly define the expected behaviour and preferences of the different classes, since only the eight-class version can be interpreted as hierarchical tiers – the other models would not naturally lend themselves to a decreasing preference function, for example.

Structure of the model

Repast provides not only a simulation backbone, but also simulation templates for common model types that require minimal adaptation. If these templates are to be used – which suits a prototype very well – there is a choice of subclassing either the Java class `SimpleModel`, or `SimModelImpl`. (Alternatively, a model could implement the `SimModel` interface directly, providing all of the basic functionality itself, but this is unnecessary.) Each is built around the basic concept used in Repast (and other simulations) of scheduling events at certain time steps, or perhaps on every step. Repast manages the scheduled events and progresses the time step automatically.

Each run of the simulation begins with a setup stage, followed by the progression of time steps, and the events associated with them. For this model, a time step was taken to represent the passing of an academic year; so on each step, a new set of children would apply for school places.

`SimpleModel` provides basic functionality, based around the core methods `preStep`, `step` and `postStep`, which are executed before, during and after each step, respectively. Alternatively, 'auto-step' mode can be used, in which each agent implements the `Steppable` interface, providing its own versions of the above methods that are automatically invoked on each time step.

However, since more advanced functionality, such as batch runs to explore the parameter space was required, the more advanced `SimModelImpl` was subclassed instead. This also allowed for a less restrictive scheduling mechanism, as it requires the model to construct its own schedule. Since the same set of actions are carried out on each time step, all that was necessary was to provide a single method to run on every step. This first cycled through all the children, asking each to apply for a school place, and then iterated through all the schools, asking them to accept as many children as they could from those who applied, and to reject the rest. These rejected children then had to begin the process again, until each had found a place. It was guaranteed that each child would eventually find a school place, since the schools are assumed to each have an equal number of places, determined by the number of children needing to be placed each year. In the worst case, a child would need to make as many applications as there were schools, but this would only usually be expected to occur for a minority of children on each step[9].

The initial model class, `SchoolChoiceModel`, was soon refactored to have two subclasses, `SchoolChoiceInteractive` and `SchoolChoiceBatch`, to more clearly separate out the code that pertained only to the batch model. It was also somewhat more efficient than checking whether the model is in batch mode before executing any operation relating to an on-screen graph.

Both interactive GUI mode and batch mode were set up to log data about the fraction of middle-class children in each school at each time step. As well as taking screen shots of

[9]The most extreme example of this would be if all children applied to the same school: then half of them could not be given a place and so they would need to apply to the second school as well. However, this is only expected to happen in the strictest class polarisation experiments, when all children strictly prefer the leading school.

the graphs created using Repast library classes, custom data analysis was possible through importing the standard format comma-separated text file into, say, Microsoft Excel.

Parameters

Repast supports parameter setting in batch mode through parameter files, and in interactive mode through a basic GUI that is created for every simulation. If the model class, in this case `SchoolChoiceModel`, uses the Accessor design pattern for methods, so that (for example) the `numSteps` parameter has associated methods `int getNumSteps()` and `setNumSteps(int)`, then those parameters are automatically made available to the user in the Repast GUI, if they are also included in the model's list of modifiable parameters.

The parameters chosen to be modifiable in the initial prototype were:

1. FractionMiddleClassPopulation – this corresponds to θ in the equations above;
2. MiddleClassPreference – this corresponds to *a* in the middle-class parental choice equation: the coefficient for how much they prefer a middle-class oriented school;
3. WorkingClassPreference – the same preference coefficient for working-class parents, which corresponds to *b* in the working-class parental choice equation;
4. NumChildren – the overall number of school applicants per year;
5. NumSchools – the number of schools to choose from; this was initially left at 2 in the simple replication of Room and Britton's model;
6. NumSteps – how many time steps to execute before halting;
7. SchoolStrategy – a binary parameter: either 'class blind' or 'class sensitive'.

Zeigler (1976, cited in Kleijnen, 1995) distinguishes between *input variables* and *parameters* to a simulation, where a variable is an attribute that can be directly observed, such as the number of schools, but a parameter cannot, so its value must be drawn from observations. An example of a parameter in Zeigler's sense would be the preference for middle-class-oriented schools held by parents. Kleijnen (1995) adds to this the concept of a *module* that can be varied between runs, giving different behaviours: for example, the module determining school strategy. However, we will use the umbrella term 'parameter' for each of these types of input to the simulation, since this is in keeping with Repast terminology.

It is in our case perhaps more useful to distinguish between parameters that are expected to significantly affect an experiment's outcome if varied, and ones that are present more out of convenience than necessity. The numChildren and numSteps parameters could be viewed as such 'convenience' parameters, since the larger the values, the more certain we can feel about our results, as the size of the sample will have increased. However, they need not be treated as rigorously as the other parameters in a parameter sweep used to investigate the model.

3.2.2 The full model

Once the initial prototype had been established to replicate Room and Britton's model satisfactorily, the technology and techniques used were assessed for suitability for the development of the full model. Despite some issues with the use of Repast library classes breaking the replicability of experiments (see section 3.3.2), which were soon overcome once identified, Repast seemed well suited not only to the development of the initial prototype, but also to the full, extendible model of the second phase of this project. Repast imposes certain restrictions on the main model class (such as having to use the Accessor design pattern, and scheduling events against time steps), but apart from that, arbitrary Java code can be used, which results in great flexibility. The agents can

implement the `Steppable` interface that structures their actions into pre-, during, and post-step phases, but this is not mandatory. The minimal restrictions that Repast does impose are therefore fully reconcilable with this project's requirements. It would also be compatible with the use of a production rule system.

Since the initial prototype did not expose any real weaknesses in the technology used, rather than starting from scratch, evolutionary development continued directly on the prototype.

3.2.3 More than two schools

The first extension to be added to the model was the ability to cope with more than two schools competing for children.

Equation 3.1 (p. 24) must be adapted to:

$$m(x) = \frac{1}{n} + a(x - \delta)$$

Equation 3.3 - fraction of middle-class parents who prefer the school with fraction middle-class x over all other schools

where n is the number of schools (previously 2), and δ is no longer the fraction of middle-class children in the population, but the fraction in the population *among those schools who have not yet rejected this child.* The reason the schools that have already rejected the child must be ignored is that if there is one excellent school A and two atrocious schools B and C, the child is likely to give $m(x_B) <= 0$ and $m(x_c) <= 0$. Then, if it is rejected from the only school it has any interest in attending, A, it has no way of choosing between schools B and C since it does not want to attend either of them to any degree (having assigned a negative preference to each). But if B and C are only compared to each other once A is out of the running, then at least one of them will be given a positive fraction/probability of choice; this is important if one is performing less badly than the other.

The children's decision procedure therefore needs to be consulted after every rejection to determine the new probabilities for choosing amongst the remaining schools. This increases the complexity of the algorithm by a factor of n.

When more than two schools can be considered, the translation of m(x) and l(x) into probabilities is also less straightforward. In the simple case of two schools using Equation 3.1, m(x) and l(x) can easily be truncated to fall between 0 and 1, giving a natural correspondence to probabilities. For example, if $m(x_A) = 4.5$ and $m(x_B) = -3.5$[10], it can be truncated to school A being picked with probability 1, and B with probability 0 without any loss of information. $m(x_A) + m(x_B) = 1$ in every case, if truncation is used (Room and Britton, 2006b). But this only holds for the two-school case, where the middle-class admissions of one school are the mirror image of admissions to the other school – a child must attend either one school or the other.

[10] e.g. if $\theta = 0.5$, $a = 20$, $x_A = 0.7$, $x_B = 0.3$

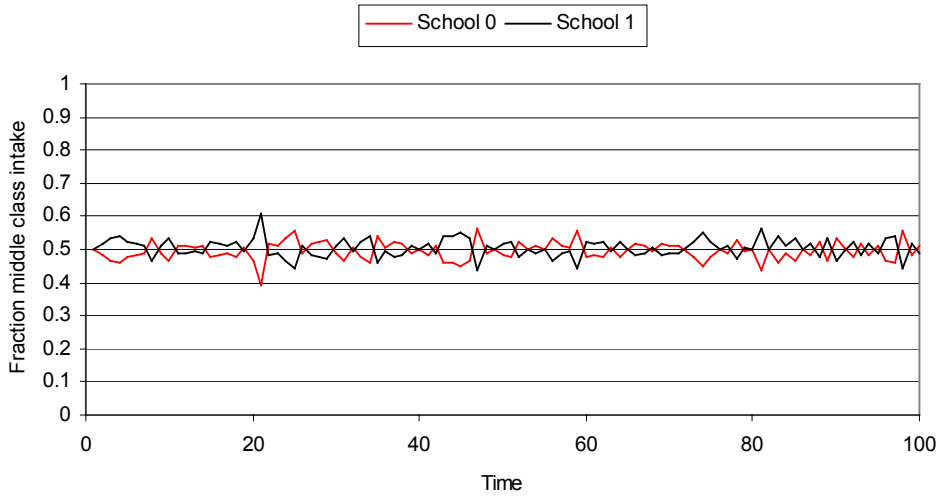


Figure 3.1 - Example of two schools necessarily 'mirroring' each others' middle-class intake fractions along $y = \theta = 0.5$.

If, however, a third school C becomes involved, values > 1 are suddenly distinguishable in Equation 3.3 since more than one can occur at once. (Any values < 0 are, however, still considered equally undesirable and are all set to 0.) For example, $m(x_A) = 2.33$, $m(x_B) = 2.4$ and $m(x_C) = -3.67$[11] cannot be truncated to 1, 1 and 0. But this problem is easily solved by converting these weightings into probabilities in the interval [0,1].

Multiple schools bring another factor into play: they necessitate the use of synchronised application rounds. When only two schools are considered, this is unnecessary, because all children rejected from one school are guaranteed to be accepted by the other: the order of applications is irrelevant. If, however, there are (say) three schools – one leading school, one middle-ground school and one failing school – then it would be unfair to give children applying to the middle-ground school only because they were rejected from the leading school the same consideration given to applicants who made the middle-ground school their first choice. Furthermore, this introduces an unwanted element of randomness where the order in which schools consider applications now has a noticeable effect on the outcome: if a school processes its applications after one or more other schools, it may also be considering some applicants rejected from those schools alongside its initial applicants. The solution is to introduce application rounds, where second-round applicants can only be considered once all schools have completed their first-round admissions process. Then, in the second round, each school gives equal precedence to each child who made that school its second choice, and so on. Unfortunately, the author only realised this late into the project, and so this change was only made to the Drools model, due to time constraints. Running multiple-school simulations in the Java model therefore still produces results that suffer from the effect described above.

3.2.4 Stochastic shocks

It would be interesting to investigate the effect of 'stochastic shocks' on the stability of schools' positions – both whether such a shock could trigger schools into a stable state

[11] e.g. if $\theta = 0.5$, $a = 20$, $x_A = 0.6$, $x_B = 0.6$, $x_C = 0.3$

they would not otherwise have reached, and whether they could be permanently knocked out of a stable state.

A stochastic shock is an additional bonus or penalty to a school's league table rating on top of the rating calculated from the school's middle-class population. A positive shock might be due to a new headmaster with a good reputation arriving, or a winning school football team, while a negative shock could be due to news coverage of an undesirable incident at the school, or a poor OFSTED report.

Three new parameters were added to the model: NumStochasticShocks, MinShock and MaxShock. MinShock and MaxShock are constraints on the minimum and maximum possible stochastic shocks, and may be positive or negative. Typically, MinShock = MaxShock in an experiment, for greater control over the effect. It is randomly decided when the shock happens, and for each school whether it is affected by the shock – a more refined model would allow full control over this as well, but for current purposes, it sufficed. Of course, reusing a random seed will replicate the sequence of events exactly, as always.

A stochastic shock could only really be expected to have an effect when the schools are in a non-polarised state: if one school normally has a 100% middle-class intake and the other receives none, a modest shock should not make much of a difference. Therefore, the schools would need to use a class-blind strategy, and the families would need to have a sufficiently high preference to result in an inequity between the schools. However, it was instead proposed to give the schools a variable preference for middle-class children, rather than the simple binary 'yes/no' preference used previously. Varying this alone should allow a degree of control over the inequity between schools. Making the schools' strategies more fine-grained improves the potential for drawing social policy conclusions from the experiments, since schools' preferences can be controlled through policy, but families' cannot (or perhaps they can be influenced, but to a lesser degree).

3.2.5 Decision rules

The most significant change made to the initial prototype was to redesign it to use decision rules instead of the more procedural approach previously adopted. This was suggested for two main reasons:

1. The model had become difficult to read, as pieces of logic were encoded in each object, so that they needed to be analysed carefully in order to discern their interactions with each other. Additionally, this logic was intertwined to an extent with less important code relating only to simulation mechanisms. Rules would make the logic more readable, especially to a non-technical domain expert.
2. Having the decision rules in a central location would make the model much more flexible and extensible – both in future, and for the continued evolutionary development during this project, which required continual refactoring.

Additionally, rules would make it easier to understand why a certain outcome was reached, as the trace of rules that were applied can be followed. This could prove invaluable in the analysis of micro-level behaviour that this project aims for.

A production rule system evokes a declarative, fact- and rule-based programming style. The fundamental concept is that 'facts' (Java objects, in this case) are 'asserted' into a processing area called 'working memory'. The condition-action rules forming the program are continually checked against the facts in working memory, and if there is a match with the rule's condition, the rule 'fires', which means that its action is run. Actions typically involve a combination of modifying facts in working memory, and running small chunks of procedural-style code (often written in Java for Java-compatible frameworks; but some extend this to integrate other languages, most commonly Lisp).

At its core, a production rule system simply consists of many rules of the form

if <condition> then <action>.

If multiple rule conditions match at once, a conflict resolution strategy must be applied. This could be as simple as following textual ordering in the file.

Initially, the suggestion was to use JBoss Rules ('Drools') as the obvious choice due to existing familiarity on the project, but some alternatives were considered in case a more suitable choice existed.

The primary constraints on the choice of rule package were that it should be:

1. Freely available software;
2. Compatible with Java;
3. Stable and reliable.

Secondary constraints were that it should be:

4. Integrated with Eclipse, this project's existing IDE;
5. Compatible with Java 1.5, to avoid having to refactor the program to make it backwards-compatible with an older version of Java;
6. Relatively fast to learn;
7. Reasonably efficient (for a rule-based language, which is necessarily slower than Java alone);
8. Open source software, for flexibility of future extension, and ease of debugging.

The options considered were as follows (excluding the most unsuitable ones):

1. JBoss Rules[12] (hereafter known as Drools, its more familiar name)

   Drools uses an object-oriented version of the Rete pattern-matching algorithm. This works by caching partial rule matches so they do not need to be re-evaluated unless object attributes change (Codehaus Foundation, 2006). This means that more memory is used, but computation is performed faster. An experimental Leaps algorithm is also supported which can identify rules to fire without evaluating the entire condition network, but it is not considered reliable yet, so it was not considered for this project.

   Drools has the following advantages:

   - It uses intuitive, minimalist syntax that can be redefined by the user to a specific domain;
   - It is established, open-source software with a large following;
   - It is exceptionally well-integrated with Eclipse;
   - It is designed for speed and scalability;
   - It supports SQL-like queries for returning data in a loop, unlike JESS;
   - The author was already acquainted with an earlier incarnation.

2. JESS[13]

   JESS, the Java Expert System Shell, also uses an adapted version of the Rete algorithm, but it looks very different to Drools as it uses a Lisp-like syntax. It is an extended version of CLIPS, a popular expert system. It has an established support base, and the project is still in active development, again like Drools. JESS is not open-source software.

[12] Available from http://labs.jboss.com/portal/jbossrules/ (accessed 22 April 2007)
[13] Available from http://herzberg.ca.sandia.gov/jess/ (accessed 22 April 2007)

3. Mandarax[14]

   Mandarax is an open-source backwards-chaining system that uses less memory than forward-chaining frameworks like JBoss Rules and JESS, but is slower as a result. Backwards-chaining is better suited to query-driven than event-driven applications (Dietrich (2003); Proctor et al. (2006) section 1.1.1). This project will only rarely need to query the working memory for facts, or work towards a goal; more usually it will need to respond to incoming events, so Mandarax does not seem like an ideal match.

   Prova[15] was recently built on the Mandarax engine to include Prolog-like syntax, but it is still a relatively untested technology

4. JLisa[16]

   Jlisa is again based on the popular Rete algorithm. JLisa is very similar to JESS; both are based on CLIPS. But while JESS provides Lisp-like syntax, JLisa makes all features of Common Lisp available. Furthermore, unlike JESS, JLisa has an open-source license. A major drawback is that there is no apparent documentation; it is a relatively recent project (begun late 2003) with only 'pre-version 1' releases.

5. OpenRules[17]

   The OpenRules system is integrated with Eclipse, and works alongside Microsoft Excel. Whilst Excel is available to this project, and used for data analysis, it would be preferable not to use proprietary software in the code itself for the sake of replicability and reuse by other researchers. OpenRules itself, however, is open source.

   OpenRules was also designed with scalability and performance in mind, but it only has a small userbase at present.

6. Algernon[18]

   Algernon, an open-source plug-in for the ontology editor Protégé, again exposes Lisp functionality, and has a complex syntax. It is flexible, supporting both forward- and backward-chaining, but again has only a small userbase.

7. SweetRules[19]

   SweetRules supports the OWL standard for semantic Web ontologies; it is oriented towards integrating semantic web rules and ontologies, which is not really a concern for this project. It is interoperable with JESS (although unlike JESS, it is open-source); and like Algernon, supports both forward- and backward inferencing, adding flexibility on both counts. As with most of the systems considered, probably because they are all relatively new technologies, it has a small userbase.

Only Drools and JESS satisfy all three primary criteria. Drools was chosen over JESS because the existing familiarity presented a strong advantage due to tight time constraints; its open source license was also attractive, although JESS is freely available for academic purposes. It is also exceptionally well integrated with Eclipse, the project's development environment, allowing interactive parsing and syntax correction of the rules as they are written, thus speeding up development. The only potential concern was that there might not be enough RAM to cope with very large numbers of agents. However, this risk applies equally to JESS, as it uses the same fundamental algorithm.

[14] Available from http://mandarax.sourceforge.net/ (accessed 22 April 2007)
[15] Available from http://www.prova.ws/ (accessed 22 April 2007)
[16] Available from http://jlisa.sourceforge.net/ (accessed 22 April 2007)
[17] Available from http://openrules.com/ (accessed 22 April 2007)
[18] Available at http://algernon-j.sourceforge.net/ (accessed 22 April 2007)
[19] Available at http://sweetrules.projects.semwebcentral.org/ (accessed 22 April 2007)

Structure of the rule-based simulation

An initial attempt at converting the simulation to use Drools involved creating a single rule base and a shared working memory, but this was soon revealed to be unsatisfactory. Instead, each agent was treated as a small expert system with its own rules and an independent working memory. This meant that the rules could be rewritten more succinctly, as interference from other agents' activities no longer needed to be considered. In particular, objects could be removed from working memory when no longer needed, rather than having to be shared between agents, which had required the introduction of 'link' ontology objects like SchoolRating to indicate that a particular Child was still considering a given School as a possible destination. The drawback was the overhead of having as many working memories as agents, but this was outweighed by the many advantages.

The rules were initially split into separate school and child rule collections, reusing the same rule set for each school or child respectively; but it would be easy to specify several different versions of a rule set for children or school agents, thereby introducing a more sophisticated type of heterogeneity among agents. Additional simulation parameters could specify what fraction of the agent population should use a particular rule set. It is also possible for more specialised agents to reuse this base rule set and add custom rules to it, as the 'co-evolutionary' versions of the agents do (see section 3.2.6).

The use of rules allowed the simulation's schedule to be simplified to remove all knowledge of how the agents interact with each other, except that children must act before schools (the old version is shown in Appendix B, section 10.1). Instead, only the following sequence was executed by the model on every step:

```
SimUtilities.shuffle(childList);
for (Child child : childList) {    // Prepare for step
      child.preStep();
}
for (School school : schoolList) {
      school.preStep();
}
for (Child child : childList) {    // Do step
      child.step();
}
for (School school : schoolList) {
      school.step();
}
for (School school : schoolList) { // Clean up after step
      school.postStep();
}
Environment.getInstance().advanceYear();
```

It would have been neater to include a call to `postStep` for the children as well, but it was left out for efficiency reasons, since that method is currently empty.

The children's `prestep` method initialises a new child, since `Child` objects are reused on every step rather than creating new objects each time, purely for the much-needed performance gain. A school's `prestep` populates its working memory with all the new applicants that might potentially apply, and its `poststep` method clears any from working memory that remain at the end of the step (children who are no longer under consideration are removed from working memory during processing). Each agent's step method simply contains a call to its working memory's `fireAllRules` method, which initially appears very elegant.

Unfortunately, it is not quite this simple: Drools' `fireAllRules` method fires all rules that have been activated by the current state of the agent, as desired, but once these have all fired, activity ceases. Rules that are activated while `fireAllRules` is still running are

added to its schedule, but those activated afterwards are not. This leads to the following kind of problem:

1) 150 children apply to school A and 250 apply to school B

2) School A calls `fireAllRules` and accepts the 150 children, leaving 50 places free

3) School B calls `fireAllRules`, accepting 200 children and rejecting 50

4) Those 50 children now apply to school A

5) School A's `fireAllRules` has already completed, and so it takes no action.

This means that additional calls to `fireAllRules` need to be inserted in the code. It is not inelegant to insert a call to `fireAllRules` in each agent's `receiveMessage` method, so that it inserts the message into working memory and then checks if any action need be taken. However, it is perhaps less obvious why in the Environment class, every time a school informs the environment that its current application round has ended, all schools have to be cycled through, calling `fireAllRules` on each. The reason is that a school might not be able to process a message immediately upon receiving it, if it is for a later application round than the current one. This mechanism could be removed if Drools supported a mode where all rules were continually executed as they were activated, which could be enabled in step, and disabled in `postStep`. However, it would no doubt reduce Drools' performance significantly, and so the current design seems preferable. Before the introduction of application rounds, the calls to `fireAllRules` had to be inserted in the children's rules; but the current solution at least places all such mechanisms in a single, meaningful place.

Again in the interest of performance, Drools does not continually poll the objects in its memory for changes; rather, it relies on the user program to inform it when something relevant has changed. This can be done either by capturing Drools' reference to the object, called a `FactHandle`, and passing this to Drools in a call to its `modify` method; or by following the Java bean-style `PropertyChangeListener` design pattern. This means that the object allows Drools to subscribe to its property change events, and is responsible for telling Drools whenever properties it deems to be of potential interest to observers change. The latter design is much more in keeping with the object-oriented program paradigm, placing the responsibility related to an object's attributes within the object itself, rather than scattering calls to `modify` throughout the program, and so it was chosen. The fact that objects monitored by Drools in this fashion must use Java bean-style 'getter' and 'setter' methods for their properties is not problematic, given that Repast already requires the use of this programming style for the main simulation model's parameters.

The objects monitored by Drools for the two types of agent are shown below:

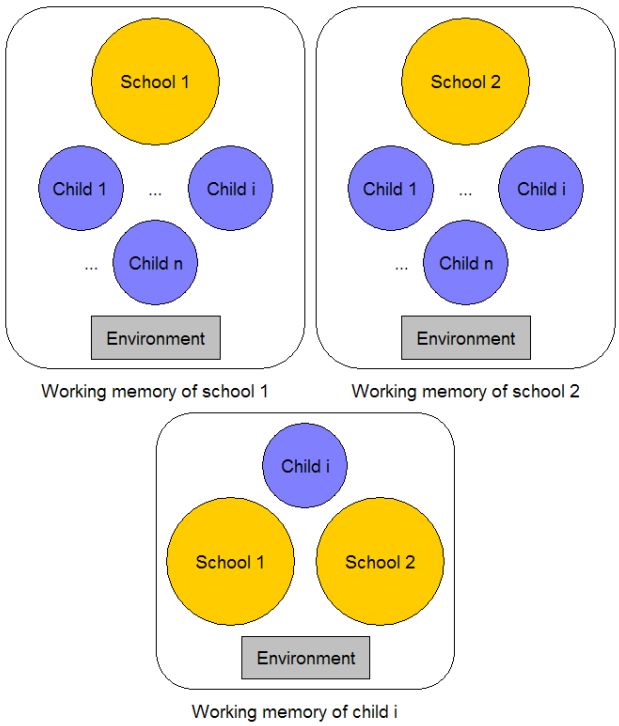


Figure 3.2 – Working memories used in the rule-based system. The top memories belong to the two schools, and the bottom one represents one of the n children in the simulation. Each working memory needs to know about the agent it is monitoring, as well as agents it interacts with directly. They also all have a reference to the Environment singleton, so the environment can be monitored for changes.

As shown in Figure 3.2, Drools has to externally monitor the agent for which it acts as an expert system, rather than being privately embedded within it in some way. This has an unfortunate side effect: any methods Drools needs to access in the object whose 'brain' it is need to have public visibility, so they are also accessible to any other object. Even though rule files can share the same package name as a Java class, this only has the effect of automatically importing all classes from that package; it does not give the rules package-level access to the classes. Thus, a fundamental principle of multi-agent systems is violated: the agents do not have control over whether they wish to carry out an action. To give over control to their 'expert system', they must also expose the functionality to any other object in the system. So, for example, there is nothing to stop a child agent from bypassing the application process and just telling the school to accept it directly. This was not a real problem in this system, because the agents are not competing with rival agents belonging to a different, perhaps malicious 'owner' who might try to 'cheat the system', so the author was easily able to avoid abusing the public visibility of the methods; but it is nevertheless a fundamental design flaw.

The rules

The rules used in the initial, model are described briefly below. The full source code is in Appendix C, sections 11.1 and 11.3. To communicate with other Drools agents, message types defined in the package `uk.ac.bath.cs.schoolchoice.ontology` are used.

1. Child rules
    a. "Research school place application"

        This rule fires when the child is in a position to apply for a school place, but has not yet made a decision. It tells the child to 'research' the schools, i.e.

calculate its preference levels for each, and then set up a probability distribution for selection.

b. "Make application for school place"

This rule fires next. The school selected by probability above is sent an 'application' message.

c. "Process rejection of school place application"

The child has received a rejection message from the school it applied to, so we must re-initialise the child so it can make a new application.

d. "Process acceptance of school place application"

The child has received a message from the school it last applied to, allocating it a place at the school. Therefore, put the child into a dormant state – it need take no more actions.

e. "Clean up schools once child has a place"

Drools can automatically tidy up its working memory by removing any remaining schools that were asserted at the beginning of the year, in readiness for the next academic year.

f. "Catch messages that were not understood"

Print an error message if the child was sent a message (ontology item) by another agent that is not caught by any other rule.

2. School rules

a. "Research school place application"

The equivalent of the child rule of the same name – all children who have applied to the school are 'researched' to determine their suitability, and a probability map is created from this.

b. "Accept school place application"

Send the applicant who was just selected by probability an acceptance message, and get ready to pick the next applicant.

c. "Reject applicants for whom we have no space"

Once the school is full, every application is sent a rejection message.

The schools could not be given equivalent rules for the children's clean-up rules 1.e and 1.f because their rules are repeatedly fired by external sources; messages or data that are not currently needed may be required at a later point. This necessitates a clean-up operation in `School`'s `postStep` method.

3.2.6 Co-evolutionary behaviour

Co-evolutionary behaviour was defined in section 2.3 as a response in, say, the child population to a particular trait of schools, followed by the schools consequently adapting their own strategy again, and so on, rather than a simple adaptation of agents to their environment. So far, no adaptation has taken place at all by either agent type, even to the environment; we now attempt to model very basic adaptation that could perhaps be termed 'co-evolutionary'. However, since the only strategies available to any agent are predetermined by those encoded in its decision rules, it could be argued that we are merely modelling adaptation at this stage, rather than true co-evolution. A more truly co-evolutionary model might employ a genetic algorithm-like technique of evolving entirely new strategies, or imitating the most successful strategies of its competitors. Nevertheless, this project will refer to such behaviours as 'co-evolutionary', since the

agent populations are continually adapting to each other, as well as the school agents adapting to their competitors.

Class blind preference

The first co-evolutionary behaviour that was added was to broaden the range of school attributes that interest families. As described by Lauder and Hughes (1999), schools that are less successful in the league tables might diversify, taking on a niche position in the market. A possible niche position for a less high-performing school would be promoting an inclusive atmosphere, i.e. their class blind admissions policy.

To simplify the co-evolutionary design, school strategies were reverted to their original restriction: they can be either class blind or class sensitive, not somewhere on the spectrum between the two. This seemed to make more sense than saying prefer a 'somewhat' inclusive school over a 'slightly' inclusive school – either way, the school is ultimately not being properly inclusive. The exact degree of discrimination is surely irrelevant to families; only its presence or absence matters.

An alternative would have been for children to look directly at the social make-up of the school, rather than indirectly at the school's strategy. However, this would have placed the two factors of an inclusive atmosphere and league table position in direct opposition, since both would have effectively been measured by the same criterion – the school's middle-class intake – with their ideal values for this criterion in direct conflict. It would also have meant that a low-performing school would not gain any advantage from a class-blind policy, as its social class make-up would be skewed to favour working-class applicants; it would rate poorly on both factors. A school adopting a class blind policy is announcing its *intention* to foster an inclusive atmosphere, and those who agree with this intention can join it, thereby enabling it to achieve its objective. If the middle-class applicants were to wait to apply until the school had already achieved a representative social mix, this mix could never come about since there would be few middle-class applicants.

To include multiple attributes in a decision, the model was refactored to use 'factors'; all factors considered by a child must implement the `SchoolFactor` Java interface. Each such factor provides a method that, for a given school, calculates the child's preference level for that school based solely on the factor in question. An equivalent design could be used for schools' consideration of applicants, but this was not included in the current design since children were given no attributes other than that of social class which a school might wish to take into consideration.

Using multiple factors meant that each had to be given a weighting. Rather than arbitrarily assuming each family assigns a weighting of, say, ⅓ to whether a school is class blind, and ⅔ to its league table performance, a normal distribution was used to create agent heterogeneity and add a touch of realism. The normal distribution's mean and standard deviation are configurable simulation parameters. As well as the weighting given to the factors, the numerical preference assigned by families to a class blind school was also designed as a configurable normal distribution, since no empirical data was readily available to suggest suitable values. (It would of course have been extremely difficult to extrapolate a numerical value from such a dataset, even if it had been available).

Having added this basic preference of families for class blind schools, rules were added to allow the agents to adapt their strategies dynamically. It seemed logical that families would be more drawn to class blind schools if the available schools were highly polarised in social class intake, and conversely that they would not care as much about the school's policy if all schools had a very similar intake, and so rules to accomplish this preference adjustment were inserted.

Schools were first given the obvious rule of a class sensitive school adopting a class blind niche if it is not performing well in the league tables. It would not necessarily want to become class blind otherwise, even if that would increase its rating with families, because it would then lose the ability to improve its league table performance above the population average. This strategy is a defence mechanism against total polarisation: a school might be able to retain some fraction of its middle-class intake if it switches to a class-blind strategy before falling too far in the league tables.

The 'mirror' rule of a class blind school becoming sufficiently highly rated in the league tables that it could afford to adopt a class sensitive strategy without losing its popularity then followed naturally. This could allow it to approach total polarisation more closely than if it remained class blind.

Finally, a rule was added for schools to try to break out of the scenario where every single schools is class blind, and none is able to distinguish itself sufficiently well to gain a permanent advantage. This is of course not morally admirable, but as discussed in section 2.1, schools aim to maximise their middle-class intake. We theorised that a possible solution was for a single school to bravely become class sensitive, in the hope that it could grasp a small initial advantage and turn it into a higher league table rating quickly enough before its poor 'inclusiveness' rating caused it to fall below the rest. However, this could easily backfire and nominate that school as everyone's least preferred school if it happened to do badly in terms of league table ratings that year, instead handing its competitors an indirect advantage through its failure.

School specialisms

School strategies and league table ratings already provide a number of interesting strategic interactions, but we wanted to include another type of school attribute that was completely independent from school league tables to investigate a fuller spectrum of multi-attribute choices. The problem with the existing two school choice factors is that they are not independent: a class blind school will have a lower league table rating than a (successful) class sensitive one. A school can either have an exceptionally high league table rating, or employ a class blind strategy, but both at once are impossible[20].

Therefore, a third factor was added to the mix that was entirely independent of the other two factors: school specialisms were introduced. This is similar to the Specialist Schools Programme in the UK (DfES, 2003a), except that questions of raising funding from the private sector were not addressed at all (this might yet be an interesting avenue to explore, since presumably more successful schools would find it easier to attract funding). It was assumed that any school can simply adopt any one of a list of possible specialisms as it pleased, but that it could only change this choice once every ten years, as continual switching would most likely be counter-productive, merely wasting resources. The UK DfES defines ten distinct specialisms which schools can apply for. The number of possible specialisms was made a configurable parameter in the simulation, since it might be interesting to see how the possible diversity (or lack of it) affects the outcome.

Each child was given an interest level for each of these specialisms according (as usual) to a normal distribution, with specialisms first ranked in a random order, and then given an interest level approximately proportional to the ranking assigned to it. This then determines how highly the child rates a school with that particular specialism. Section 3.3.7 uses the example of adding the school specialism factors to show how the model can be extended in detail.

[20] Unless of course working-class children were to suddenly prefer class sensitive schools over class blind ones, while middle-class children continued to favour high-performing class blind schools as expected, but that would make no sense, so it was not explored.

Section 102 of the School Standards and Framework Act 1998 declares that any specialist school may select up to 10% of its intake based on the applicant's aptitude in the school's specialism (DfES, 2003b). This would add an extra twist to the simulation, giving child agents more than one attribute of interest to schools. Unfortunately there was no time to add this to the simulation, but `ChildFactor`s could easily be introduced that worked analogously to the existing `SchoolFactor`s. The DfES also states that a school whose performance is declining is unlikely to gain approval for its application for specialist status; this could have provided some interesting interactions with other factors, had there been time to incorporate this extra condition.

The rules added to the system for schools to decide upon their specialism were twofold. Firstly, if a school has a more successful competitor who has chosen the same specialism, it will switch to another type if possible, since that might open up a new corner of the market, and it is not gaining any advantage from its current specialism. Secondly, a ruthless rule was added for successful schools: they can choose the same specialism as a less successful competitor in order to steal that school's niche. This would leave the other school with the same poor league table performance as previously, but bereft of the unique offering it had. All children who previously preferred that school because of their personal interests will now join the other children who preferred the leading school precisely because it was so successful. All the children who would have chosen the leading school because they were interested in its old specialism will now rate the leading school less highly; but the specialism attribute has effectively been eliminated from the equation (being equal for both schools), so it does not matter. The interactions between more than two schools would be more complex; it remains to be seen whether the rule still pays off in that scenario.

Since the children change every year, and therefore the applicants' favoured specialisms do as well, there is no advantage in schools trying to cater to specialisms they perceive as being popular. It might be more realistic (and more interesting) if trends were added to children's interest levels, where perhaps their older siblings and other members of the community could conceivably influence their interest levels to positively correlate with the existing interests in that area. However, this would be most useful if data could be found on how high such correlations are. Again, the basic problem of quantifying such a subjective issue as 'interest' in a subject would be a major issue, even if the ideal dataset were available. As it is, the children's linear decrease in preference among the ranked specialisms is no doubt inaccurate; but it seemed preferable to use an easily comprehensible preference assignment than to guess at a more complex function that would inevitably also be wrong.

Structure of working memory

After adding the co-evolutionary behaviour, the agents' working memories needed to consider more facts, as shown below:

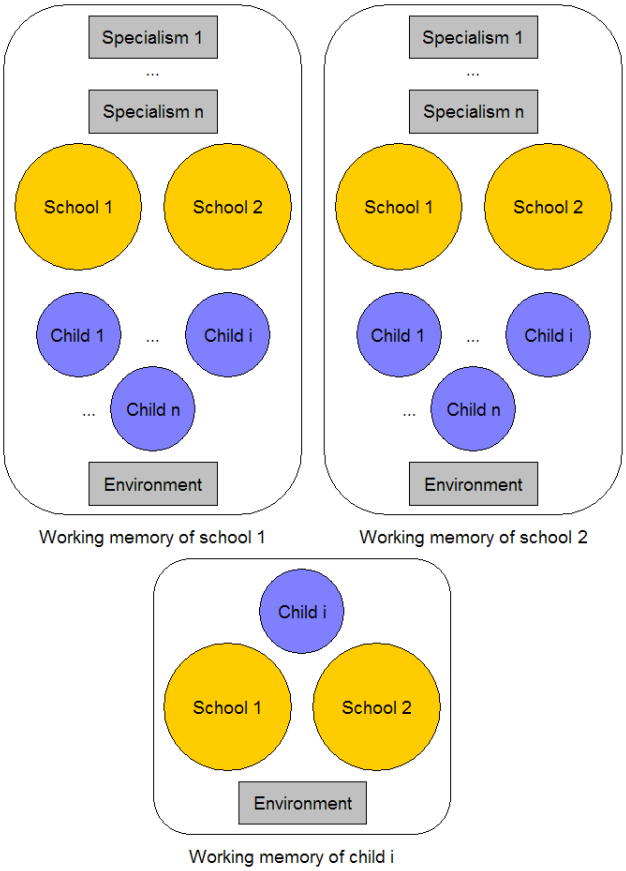


Figure 3.3 – Working memory contents of the co-evolutionary model. This expands upon Figure 3.2. The children's working memory remains unchanged, but each school now needs to know about its competitor(s), as well as the domain knowledge of what specialisms it could potentially adopt. Note that the children are much less sophisticated than schools: they do not reason about their competitors. Schools, however, need to take account of their competitors' strategies and specialisms.

The rules

The ruleset added to the basic rules outlined in the previous section are summarised below (continuing the previous numbering to give each rule a unique number, used in the experiments). The rules' source code can be found in Appendix C, sections 11.2 and 11.4.

3. Co-evolutionary child rules
    a. "Increase preference for class blind schools due to polarisation"

       Since the schools the child is considering are severely polarised by social class (the leading school must have a middle-class intake of 170% or more of the population average), the child is told to inflate its natural preference for class blind schools since the situation is more extreme than usual.

    b. "Decrease preference for class blind schools due to typical distribution"

       The differentiation between schools is not very significant – each school has a middle-class intake that is within 15% of the population average – so the child is asked to lower its natural preference for class blind schools, since

even class sensitive schools are currently well-balanced. This rule is the complement to the previous one.

c. "Remove class blind factor if all other schools are class blind"

d. "Remove class blind factor if no schools are class blind"

These two rules tell the child to remove its class blind preference factor from consideration, since it would make no difference: all schools would be equally preferred according to this factor. The weighting previously given to this factor can now be redistributed among the other factors, giving more significance to the differences between the schools.

e. "Remove specialism factor if all schools have the same specialism"

This rule is in the same spirit as the two previous ones: if there is no difference between schools' specialisms, the child should not take its interest levels into account when evaluating the schools. This could also occur if the schools had no specialisms.

4. Co-evolutionary school rules

a. "Adopt class blind niche"

If the school is not doing very well in the league tables (10% or more below the population average), and its successful competitors are mostly using class sensitive strategies, it should switch to a class blind strategy to adopt a niche market. Children may overlook its poor league table rating in light of its inclusive admissions policy.

b. "Popular enough to become class sensitive"

This is the converse of the previous rule – if a class blind school is doing well in the league tables (10% or more above the population average), it should gamble that children's class blind preferences are not extremely high, and switch to a class sensitive strategy to allow it to climb even higher.

c. "Everyone else is class blind and we are all doing roughly the same"

This rule was added to break schools out of the stable state in which all schools are class blind, and no leader can emerge since parental preferences are too low. One school takes the risk of becoming class sensitive so it can concentrate on accepting as many middle-class applicants as possible. This rule could backfire if the children's preference for class blind schools is too high, but it also opens up the opportunity for the school to excel where it could otherwise only have remained average.

d. "Change specialism to gain edge over leading school"

If the school has a competitor who is rated higher in the league tables, and shares the same specialism, it should switch to a different specialism in the hope of attracting a niche market for that specialist subject instead. Otherwise, any children with a high interest in the subject would apply to the higher-rated school.

e. "Change specialism to that of lesser school to take away its edge"

This is again the converse of the previous rule; a 'sabotage' strategy. If the school has an unsuccessful competitor with a different specialism, it should switch to that specialism as well in order to destroy the school's specialist niche. All children who would previously have favoured that school now apply to this school instead since it is higher-rated. The school can take the risk of losing its own specialist niche market since it already caters to performance-oriented children anyway.

3.3 Implementation

3.3.1 Parameters

The following parameters are present in the final model:

1. Fundamental parameters
    a. NumSteps – how many years the simulation should run for.
    b. NumSchools – the total number of schools used in the simulation.
    c. NumChildren – the total number of children applying for school places every year. The number of school places will be determined from this number, giving an equal number of places to each school. If the number of children does not divide evenly by the number of schools, the number of places in each school is rounded up, so a school could be left with up to numSchools - 1 unfilled places every year.
    d. MiddleClassPreference – the preference coefficient given by middle-class children to high-performing schools. This is equivalent to *a* in Room and Britton's model.
    e. WorkingClassPreference – the preference coefficient given by working-class children to high-performing schools. This is equivalent to *b* in Room and Britton's model.
    f. SchoolPreferenceForMiddleClass – how much schools prefer middle-class children. A preference of 1.0 is equivalent to a class-blind strategy, while the maximum integer value (shown in the screenshot) is equivalent to a fully class-sensitive policy.
2. Basic extensions
    a. InitialTwoSchoolInequity – the initial league table inequity two schools begin with at the start of the simulation. This only takes effect if only two schools are involved.
    b. NumStochasticShocks – how many stochastic shocks should be injected. They will occur in a random year, and hit random school(s) within that year.
    c. MinShock – the minimum value stochastic shocks can take.
    d. MaxShock – the maximum value stochastic shocks can take. Like 2c), only valid if NumStochasticShocks > 0.
3. Co-evolutionary behaviour
    a. NumSchoolsClassBlind – the number of schools out of NumSchools that use class blind allocation strategies. For these schools, SchoolPreferenceForMiddleClass will not be used. This provides a crude mechanism for initialising schools with different strategies.
    b. ClassBlindPreferenceMean – the mean used in the normal distribution from which a child's preference for class blind schools is picked. This is only valid if co-evolutionary behaviour is activated.
    c. ClassBlindPreferenceStdDeviation – the standard deviation used for the same purpose as 3b).
    d. LeagueTableWeightingMean – the mean used in the normal distribution used to generate the weighting given by a child to schools' league table positions. The remaining weighting from 1.0 is divided equally between the other factors.

e. LeagueTableWeightingStdDeviation – the standard deviation used for the same purpose as 3d).

f. NumSpecialisms – the distinct number of specialisms between which schools can choose.

Appendix B, section 10.1 gives an example of a batch parameter file, and instructions for running the simulation.

3.3.2 Replicability of experiments

Replicability is a vital attribute of any experiment. The results of an experiment cannot be trusted unless they can be replicated under the same conditions. For a computer simulation, this means that any stochastic elements must be deterministic. Repast allows this to be accomplished through its encapsulation of various functions from the Colt[21] pseudo-random number library. A pseudo-random number generator will always generate the same sequence of 'random' numbers if passed the same 'random seed' for initial configuration.

In addition to the custom parameters detailed above, Repast automatically makes the RandomSeed parameter available to the user. This should be controlled carefully in experiments.

Replicability is violated if any randomness that does not depend on the random seed is introduced, such as use of standard Java's `java.util.Random` class. That is easy to avoid, but randomness introduced by an unordered data structure, like a hash table, whose contents are iterated through sequentially, is harder to detect. Such an ordering will not necessarily stay constant from one run to the next. Strangely, the Repast libraries themselves make use of this coding practice on occasion, and so those particular library classes could not be used. For this model, only `uchicago.src.sim.network.Uniform-Reinforcement` had to be replaced by a custom class, `DeterministicProbabilityRule`. This did have the advantage that the project code was simplified, since only the basic functionality of `UniformReinforcement` was needed, and so 'dummy' arguments to the method calls could be removed.

The fact that Repast library code cannot be relied upon to act deterministically, despite the developers' obvious recognition of the need for repeatable results (evidenced by the provision of the RandomSeed parameter, and remarks in their tutorials[22]), is disappointing. It appears to break the concept of encapsulation that Repast library classes cannot safely be used without first reading the source code to check for violations of determinism.

3.3.3 Ordering within the rule body

Another problem encountered was that the ordering of multiple actions within a rule body (or 'consequence') was very sensitive. One might naïvely assume that in a declarative environment like a production rule system this would not be the case, but the problem lies in the fact that one rule firing may trigger another – and this can occur before the original rule has finished executing. So if a rule body contains two actions, each of which modifies a property of an object, another rule may fire as soon as the first property was changed, even though the second property modification would have suppressed it again. The problem with this is that the side effects of an action are not always immediately

[21] "a set of Open Source Libraries for High Performance Scientific and Technical Computing in Java" – available from http://dsd.lbl.gov/~hoschek/colt/index.html (accessed 11 March 2007)

[22] The definitive random numbers tutorial for Repast is available at http://repast.sourceforge.net/how-to/random.html (accessed 23 April 2007)

apparent, especially when one of a cohesive group is considered in isolation. A synchronisation mechanism such as that used in database transactions would have been of help here. Fortunately, while this was an issue during evolutionary development, refactoring simplified the final version of the simulation so that it was no longer a problem. Nonetheless, future extensions of the simulation could complicate the design to a point where it became an issue again.

It should be noted that this would not be an issue if the property change listener design pattern was not used to notify Drools of changes to agents' properties (see section 3.2.5). If this were replaced by a Drools call to `modify(agent)` at the end of each rule's body, the ordering within the rule body would no longer matter. However, additional calls to `modify` would need to be scattered throughout the Java code, which we wished to avoid.

3.3.4 Memory leaks

Running many simulations back-to-back in batch mode, sometimes for days on end, meant that even a small memory leak, which might ordinarily go undetected, was catastrophic. A large portion of the debugging effort was therefore spent on this type of bug, despite Java's provision of garbage-collection (these bugs were typically due to static references, which cannot be garbage-collected). One in particular was noteworthy; it appeared when the Java prototype was converted to use Drools. Drools supports the use of a `PropertyChangeListener` helper class, which informs the working memory when any Java bean-style property thought to be of potential interest has changed (see section 3.2.5). This is a common design pattern, and so an implementation of `PropertyChangeListener` is provided in the standard Java library code (`java.beans.PropertyChangeSupport`). This class was found to not be garbage-collectable. Fortunately, an open source alternative was available[23] since the problem had been encountered by others before. However, the trade-off for obtaining a memory-safe class is that it is no longer serializable. This issue would need to be addressed if the model were to be developed into distributed software.

3.3.5 Performance

The performance of Drools programs can vary widely depending on whether the rules are written with efficiency concerns in mind (Proctor et al., 2006, section 7). Caching the compiled rule packages removes much of the initialisation overhead of using one working memory per agent. However, it was the performance of the agents during the simulation run that proved a serious issue for this project. While the Java model took 1 minute and 24 seconds to run a basic two-school scenario for 200 years[24], the Drools model took 38 minutes, or 53 minutes if all co-evolutionary behaviour was enabled. The use of Drools clearly adds huge overheads to the simulation: a working memory needs to be set up for each of 402 agents, each of which maintains references to many of the other agents and objects, and handles its own sophisticated rule-firing mechanism. The co-evolutionary behaviour causes an additional drop in performance because its rules have greater complexity than the standard ones. Drools cannot cache some of these more sophisticated conditions; this has a big impact on efficiency.

The class blind niche adoption rule is the worst culprit: it includes a Drools function call in the rule's condition, for checking whether 'most' of the school's competitors are class sensitive. The woolly concept of 'most' is not one that can be expressed using Drools

[23] Under IBM's Common Public License; available from Simmons(2004).

[24] These three tests were each run on alis.cs.bath.ac.uk, a Linux machine with four 2.66GHz CPUs and approximately 1.5GB of RAM. Approximately 34% of alis' memory was used in the most demanding test, and one of the CPUs was used at around 99% capacity throughout, to the best of the author's knowledge. A distributed program would have been able to take advantage of alis' multiple CPUs, which would have led to a large performance improvement.

primitives, and so a Drools function must be used to enable the condition to be defined through arbitrary Java code. Naturally, the result of this function call cannot be cached by Drools, since it has no way of knowing what data is referenced, and therefore when the return value might change. It might, however, be possible to express a 'most' relationship using the new Drools conditional element 'accumulate', which can be used for set operations (JBoss, 2007).

The other major factor in the simulation's poor performance is that the addition of co-evolutionary behaviour required schools' working memories to hold references to the school's competitors, as well as to the school itself. To identify the school agent for whom the working memory is acting, a global variable 'myID' is used, which is matched to the schools' ID property. Use of 'myID' would require a check like this in every rule:

```
school : School( placesLeft > 0, ID == myID).
```

Unfortunately, Drools does not currently support the use of global variables in this manner. There is no apparent technical reason why this cannot be done. The rule engine already assumes that global variables remain constant throughout execution, so it cannot be a question of how Drools could tell that the variable had been modified; it seems that it has simply not been implemented yet. Instead, the check must be wrapped in a call to `eval`:

```
school : School( placesLeft > 0, ID == myID)
eval(school.getID() == myID).
```

Drools cannot cache anything inside an `eval` statement, so this places a huge run-time overhead on the rule engine.

This burden was propagated to the basic school rules, which, being sometimes used by co-evolutionary schools, needed to be able to cope with several different schools being present in working memory. Fortunately, the check could be bypassed in many cases: namely, where the school was considering a child's application. The application message is only sent to the school it is being made to, which it references directly in a property, so a rule containing e.g.

```
school : School( placesLeft > 0)
Application(appRound : applicationRound, schoolInvolved == school)
```

does not require an additional check against `myID`, since the only school in working memory matching the constraint of being referenced by the application will be the school which the rules are operating on.

The negative impact of rule conditions that cannot be cached can be lessened by placing them last in the list of rule conditions; then they will not be evaluated unless all other conditions match. More generally, rule conditions should always be ordered so that the condition least likely to match is listed first; this will save unnecessary evaluation of the other conditions whenever it does not match. When the rules were refactored in order to improve performance, significant gains were found here.

The Drools manual (Proctor et al., 2006, section 7) suggests that disabling the indexing done automatically can help performance in some cases, although the savings are generally seen in memory usage rather than in speed. However, typically indexing should be left enabled, as it is of course designed to improve performance in most systems. Left- and right-indexing of node matches can be enabled and disabled independently. We found that disabling left-indexing brought no performance gain; and unexpectedly, disabling right-indexing caused the system to no longer function correctly! The less favoured school was not filling all its places, leaving some children unallocated; this indicates that its application acceptance rule was not firing in the second round of applications. This could mean that the rules inadvertently rely on some aspect of right-indexing that they should not, strictly speaking. To determine whether this was the case would require an in-

depth investigation of Drools' indexing mechanism, but this discovery was only made at the end of the project. However, the Drools development team does not believe this is possible[25]. In any case, the effect of disabling indexing various enormously from rule set to rule set, as it is only a 'tweak' for unusual rule combinations, so it would be of little use in a project where the rule set itself is an experimental parameter, and keeps changing.

Were this project to be continued on a larger scale, the simulation's poor performance would need to be addressed more seriously. The forthcoming release of Drools does claim that it is "much faster than before, and uses less memory"; however, it is currently "unstable" and was therefore not tested (JBoss, 2007).

3.3.6 Tracing and logging

Any non-trivial simulation will soon become so complex that a detailed analysis of its output must examine exactly which behaviours or rules were fired when; to guess using intuition is insufficient, because it is easy to overlook an interaction between rules. The next version of Drools will offer debugging facilities for examining working memory and the activation stack directly; this would be useful, but it is not sufficient. Since the experiments take so long to run, each should have an associated log of which key rules were used, and at what time.

Drools offers an 'audit' facility that produces a log of which rules were fired; it monitors one working memory at a time. The volume of data this produces is immense, although filters can be set up; what is perhaps more problematic is that a log is created for each working memory in isolation, whereas we are also interested in which agent acts first, and how their actions affect each other. For example, if one school changes its strategy, a competitor may also change its strategy, in response. The output is also not especially human-readable (a small selection is shown in Appendix B, section 10.3).

Instead, we used our own basic logger that writes both to the console and to one single file per run. A print statement can be put inside any rule in which the researcher is interested; like any print statement, it can of course also reference other variables if desired. This allows the exact order of events to be tracked, and to be separated by academic year. It was sufficient to understand the underlying behaviour in the experiments undertaken in this project. This approach was also more efficient than giving each agent its own logger that has to constantly monitor working memory.

3.3.7 Extendibility

The Java model is not especially extendible, but the Drools model supports evolutionary development very well: it is generally very straightforward to add new functionality. If the system is simply to behave in a different way using existing data, it is likely that only the Drools rules will need to be modified. Since the rules are re-compiled every time the simulation is run, the rules files can be edited as plain text without even needing to recompile the Java application, making it suitable even for non-programmers.

Adding new functionality does, of course, require modifications to be made to the Java model as well. By way of example, here is a description of the steps taken to add a new attribute to schools that children take into account during school choice: school specialisms, to be matched against children's interests.

1. Add a new parameter to the simulation: the number of possible specialisms schools can adopt. This merely requires adding a variable to the main model class, `SchoolChoiceModel`, with Java bean-style getter and setter methods, and adding the

[25] See the ongoing support discussion at http://www.nabble.com/Disabling-indexing-causes-error-tf3668001.html#a10254888 (accessed 30 April 2007)

name of the parameter to the list of parameters in the `getInitParams` method. Repast now automatically adds the parameter to its list of configurable values, both in the user interface and via parameter files.

2. We must now give each school a specialism. We decide that specialisms are a type of co-evolutionary behaviour, and so we add this to `School`'s subclass `Evolutionary-School`, rather than to `School` itself. A third alternative would be to create an entirely new subclass of `School`, with its own 'add-on' rule file that extends the base set of rules like the co-evolutionary rules do. Since we may later wish to expand upon the concept of a specialism, we create a `Specialism` class rather than just using an integer identifier, which would suffice at present.

   `EvolutionaryChild` is assigned an interest level in each specialism, according to a normal distribution centred on decreasing values for each specialism in the list. To increase heterogeneity among the agents, we must randomise the list of specialisms before assigning preference values. Since interest levels are assumed to be different for each child, we reset them every year in the child's `preStep` method (remembering that the `Child` objects are reused every year for efficiency, but represent a new child each time).

3. Give `EvolutionaryChild` the ability to consider a school's specialism, by creating a new subclass of `SchoolFactor`, `SpecialismFactor`, with a simple decision procedure of querying the child for its interest level in a given specialism. This new factor is now added to the weighted list of factors in `EvolutionaryChild`'s `setupSchoolChoice-Factors`, It seems that the league table performance is probably still the most important factor of the three, but the exact importance is arguable, so the parameters for its normal distribution (mean and standard deviation) are added as simulation parameters as in step 1). For now, the remainder of the weighting is divided equally between a preference for class blind schools, and the school's specialism.

4. Optionally, new rules can be added to work with the new data. Here, rules were added for schools to change specialism under certain circumstances, and for children to reassign the weighting they gave specialisms if all schools have the same specialism. In order for Drools to know about data in the model, it must be 'asserted' into the appropriate working memory; agents have the convenience methods `assertConstant` and `assertVariable` for this purpose. `assertConstant` should be used for efficiency purposes if the data is immutable or any change is irrelevant; if `assertVariable` is to be used, the data class must support property listeners as described in section 3.2.5.

3.4 Testing

As for any simulation, the primary form of testing was through validation of experimental output. The fundamental aspects of the model could be validated against Room and Britton's original model, as detailed in chapter 5, but the extensions could not be tested as thoroughly, having only the author's predictions and common-sense reasoning to be compared against. All features of the model were subject to a sensitivity analysis/parameter sweep as discussed in chapter 4, in the hope of uncovering anomalous behaviour, but due to the vast parameter space, only a small subset of possible parameter combinations could be tested. The model can therefore certainly not be said to be 'correct'; only that it conforms to those tests that were chosen. However, any non-trivial simulation suffers from this criticism to an extent.

Since the model was constantly evolving during the project, regression tests were used throughout to verify that the addition of a feature had not compromised previously

working functionality. Due to the nature of the model, which contains many stochastic components, these tests could not be automated by test harnesses such as JUnit which check against expected results, such as might be used in conventional software development. Even a minor change to the program generally caused the output to change, even using the same random seed, because the pseudo-random numbers were used slightly differently in the two versions. The overall pattern of the output therefore had to be manually checked, which limited how often such tests could be performed, especially given the system's long running times. The most comprehensive set of tests of this kind were run when the system was converted from the initial Java prototype to the Drools version, as detailed in section 9.3. These were re-run many times throughout the development of the Drools model, and then detailed in this report for the final version of the program. The fact that the definitive set of regression tests could only be run once all development had finished, yet the program was designed to progress incrementally, meant that the parameter space could not be covered as thoroughly as it should. Instead, the most representative tests were chosen.

4 Design of Experiments

4.1 Experimental variables

Independent variables

For the initial prototype, the only direct parameters (defined in section 3.2) thought to be significant to the experimental results were:

1. FractionMiddleClassPopulation;
2. MiddleClassPreference;
3. WorkingClassPreference;
4. SchoolStrategy

For the full model, the following additional parameters (defined in section 3.3.1) were viewed as significant:

1. SchoolPreferenceForMiddleClass – this replaces the SchoolStrategy parameter
2. NumSchools
3. InitialTwoSchoolInequity
4. NumStochasticShocks
5. MinShock and MaxShock
6. NumSchoolsClassBlind
7. ClassBlindPreferenceMean
8. ClassBlindPreferenceStdDeviation
9. LeagueTableWeightingMean
10. LeagueTableWeightingStdDeviation
11. NumSpecialisms

Program behaviour – specifically, the presence or absence of Drools rules – presented another kind of parameter that could be manipulated.

Dependent variables

In the *class blind* case, the dependent variables were:

1. Whether schools take on stable, differentiated positions with regard to the fraction of middle-class intake;
2. If so, how soon this occurs; and
3. How large the inequity is (z^* in Room and Britton's paper).

In the *class sensitive* case, the dependent variables were:

1. Whether the schools take on completely polarised roles where the leading school accepts as many middle-class children as exist and it has places for, and the 'second-rate' school receives only those middle-class children for whom there is no space in the leading school;
2. If so, how soon this occurs.

However, in each experiment other aspects of the output could take on significance, such as patterns observed in some cases but not others. This type of qualitative analysis could not be predicted in advance, especially for a model of a complex system.

Confounding variables

The choice of random seed introduces 'noise', so several runs of an experiment should be run in order to gain confidence that the results obtained are not dependent on the particular sequence of random numbers used. Comparison of these runs involved looking at overall patterns and trends rather than local 'random noise' that would be impossible to replicate with a different random seed. An achievable number of repeat runs in a project of this scale was set at 100, although of course more repeat runs would increase the reliability of the results. It was initially set at 10 with the intention of repeating most experiments 10 times, but this was found to be unworkable, both due to the large number of experiments with long running times, and the large variability that was often found within only 10 runs. Instead, it seemed preferable to choose some representative runs and repeat them 100 times to ascertain more accurately the variability present for each distinct type of experiment. The same random seed, 1171139146239, was used in every other experiment unless stated otherwise.

The number of children could act as a confounding variable if there are too few agents to form a representative sample. Similarly, the number of 'steps' or years the simulation runs for could conceal interesting or anomalous behaviour if set too low. These issues are addressed in the next section.

There was also a risk of even minor program changes acting as a confounding variable. This is because when adding, removing, or even just reordering uses of the random number generator, different results should be expected. The results would not be expected to differ to any degree greater than that of using a different random seed, of course. Nevertheless, it was important to be aware of this source of extraneous randomness, and to eliminate it where possible by keeping the program constant. Since the project progressed in an evolutionary manner, with experiments being run in parallel with development throughout, it would have taken too long to repeat every single experiment right at the end of development to ensure they all used the same program. A workable compromise was to run all experiments up to 6.1 (inclusive) against the final version of the Java model, and all results dependent on the Drools extensions against the final version of the Drools model, allowing the first set of experiments to run while development progressed on later extensions. A thorough validation of the Drools model against the Java model was also performed in experiment 6.2, to give an indication of the level of agreement between the two sets of experiments.

4.2 Sensitivity analysis

The first set of experiments to be conducted should be a form of sensitivity analysis, to give confidence in the model: in other words, to confirm that it is not unreasonably affected by minor alterations in parameters. The caveat that it is modelling a complex system, so entirely predictable results would also be a cause for concern, should however be borne in mind.

It is unfeasible to investigate the full parameter space of any nontrivial model. At least the digital nature of the simulation means that we cannot consider the infinite domain of real numbers (only those that can be represented by the computer); but even so, an intractable number of different combinations would result if every possible parameter value were combined with every other.

Instead, the experimenter should rank the parameters in order of expected significance, and initially undertake a relatively comprehensive investigation of the most important parameter's space. Other parameters should be set to values felt to be representative or 'sensible' based both on intuition, the existing analytical study of Room and Britton's model, and a little informal experimentation with our model. This should reveal

'sensitive' areas where that particular parameter has a significant effect on the simulation. It could be, for example, that a parameter has the same impact for all values >= 10. This would allow us to prune the parameter space search tree of all branches on which that particular parameter is greater than 10, leaving us with a representative test where it is equal to 10.

Next, the parameter assessed as being second in importance should be varied alongside the first parameter, keeping the first parameter within the 'sensitive' set of values identified for it previously. Next the third parameter will be varied alongside the first and second, and so forth.

This approach has drawbacks: the ranking of parameters is subjective and prone to expectation bias. More significantly, our assumption that parameters producing the same experimental result can be compressed into a single parameter, representative of that result, ignores the fact that in a complex system, unexpected interactions may occur between parameters. However, time constraints prevent a more thorough method being used, such as the automated parameter search techniques described in section 2.6.

The question of how fine-grained the parameter increments should be remains. A simple linear increase by a small, constant value would be one option; or alternatively, a logarithmic scale could be used. Logarithmic variations suit two types of parameter especially well:

1. A parameter whose domain starts at a constant value (say 0) and increases into infinity, since the larger the parameter becomes, the less variation between intervals one would expect;
2. A parameter that is defined by a strict upper and lower limit (say 0 and 1), but has a particular value of interest, around which interesting results are expected to cluster.

Those types of parameter appear in this model among the significant set: parental preference (of either class) for middle-class schools is of type 1, and the fraction of middle-class children in the population is of type 2, with the most polarised behaviour expected to take place at 0.5 according to the analysis of Room and Britton's model. Therefore, an efficient set of values for analysing this type 2 parameter was the series

0.18 0.34 0.42 0.46 0.48 0.49 0.5 0.51 0.52 0.54 0.58 0.66 0.82

that centres logarithmically around 0.5, the most interesting value.

Alongside SchoolStrategy, which is a simple binary value in the initial prototype, those three parameters form the set of significant parameters in the model. In the more advanced model, SchoolStrategy was converted into a type 1 parameter, SchoolPreferenceForMiddleClass, which was expected to produce similar behaviour to the parental preferences, so logarithmic parameter adjustment seemed appropriate in all cases.

The SchoolStrategy parameter alters the significance ranking of the remaining parameters, so the two cases 'class blind' and 'class sensitive' were considered entirely separately. The ranking of the remaining three parameters felt to be most appropriate was as follows:

1. In the class blind case:
    1. Middle-class parental preference for schools
        - Set working-class preference = ½ middle-class preference
        - Set fraction middle-class population = 0.5
    2. Working-class parental preference for schools
        - Set fraction middle-class population = 0.5

3. Fraction middle-class population

1. In the class sensitive case:
    1. Middle-class parental preference for schools
        - Set working-class preference = ½ middle-class preference
        - Set fraction middle-class population = 0.5
    2. Working-class parental preference for schools (predicted by Room and Britton's model to be irrelevant in this case)
    3. Fraction middle-class population

Even though the fraction of the population that was middle-class was held to be more important than the working-class parental preference, it was considered preferable to eliminate the need to consider the working-class preference parameter as early on as possible, which is why it was investigated earlier. In retrospect, this was probably a mistake, and the order of the last two parameters should have been reversed. Although informal tests (as well as common sense reasoning) had indicated that the working-class parameter also had no impact on unusual values of the middle-class fraction, this was not investigated formally as it should have been.

The greater the number of children used, the more accurate the experimental outcome; or rather, the closer it would be expected to lie to the results of the mathematical model, since statistical anomalies would be averaged out in a larger dataset. However, large numbers of children would require more computing power, and not be as realistic. In fact, statistical anomalies do occur in real life; therein lies some of the interest of empirical experimentation. It seemed from discussions and experimentation that 200 children per school, per year was a reasonable number. An experiment investigated the effect of varying this number.

The number of steps required varied from experiment to experiment – in the class blind case, 500 steps were initially chosen to ensure that no unexpected behaviour emerging later on in the simulation was being overlooked. However, the rapid emergence of the final pattern, and its obvious stability, soon led to this being reduced to 200 steps; even this seemed 'generous'. In the class sensitive case, the most appropriate number of steps was less clear, since in the most borderline of cases, we might wait indefinitely for polarisation to occur. The somewhat arbitrary cut-off point of 1500 years was chosen, since most activity of interest seemed to occur within 1500 years even in the borderline cases. A few experiments required this number to be increased; this was judged on a case-by-case basis after initially using 1500 years. Of course, no social policy would even contemplate the adoption of such a long-term view, but we nevertheless investigated the model to this depth to enable us to better understand its behaviour.

5 Experimental Results: Replicating the original model

The initial Java-based prototype must first be shown to replicate Room and Britton's findings, as a basic form of validation. This was accomplished, although section 5.1.1 details a discrepancy that we argue necessitates an adjustment to Room and Britton's model. All of the experiments involve only two schools, as in the original model.

Throughout this chapter and the next, "middle-class preference" is used to refer to the preference of middle-class families for schools rated highly in the league tables (*a* in Room and Britton's model), and "working-class preference" equivalently (*b*).

5.1 Class blind case

5.1.1 Sweep of middle-class – working-class preferences for successful schools

As explained in section 4.2, middle-class families' preference for high-performing schools is taken to be the most significant parameter. Experimentation showed that logarithmically increasing values of 0.01, 0.02, 0.04, 0.08, 0.16, 0.32 and 0.64 all produced essentially the same effect, being too low to produce any significant differentiation. Values from 1.28 onwards produced increasingly greater differentiation. Therefore, the middle-class preferences chosen for this experiment were 0.64, 1.28, 2.56, 5.12, 10.24, and 20.48. Each was paired against the working-class preferences 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12, 10.24, and 20.48.

Room and Britton assume that the working-class preference is always less than the middle-class preference, so pairings that do not satisfy this constraint cannot be validated against their model (they are also not realistic, as Room and Britton argue, but nevertheless interesting to consider briefly).

As was also discussed in section 4.2, the fraction of middle-class children in the population was kept constant at 0.5; each school was given 200 places; and the simulation was run for 500 years. The experiment produced three kinds of result:

a) If the preference values were too low, or identical, no differentiation occurred:

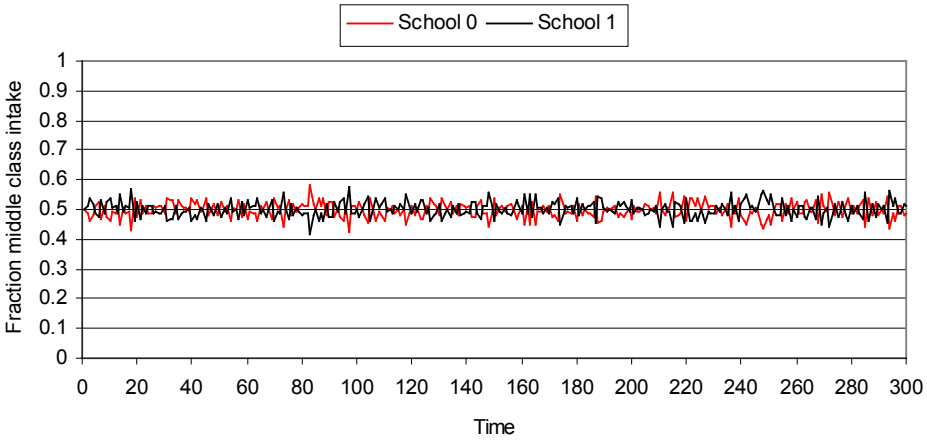


Figure 5.1 – Class blind schools with middle-class preference 0.64, working-class preference 0.01. These preferences are too weak to produce differentiation

b) If the middle-class preference was suitably high, and the difference between it and the working-class preference large enough, the two schools divided into a leading and 'second-rate' role:

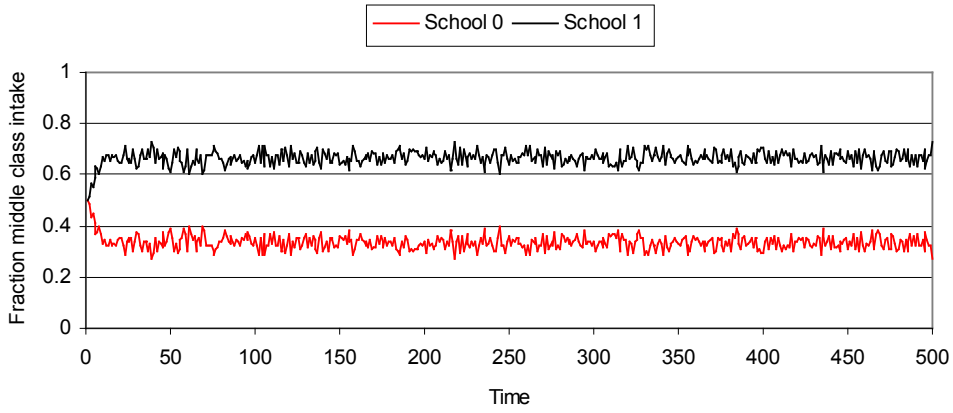


Figure 5.2 - Middle-class preference 5.12 and working-class preference 0.01 for middle-class schools produces a marked differentiation between the schools' middle-class intake

c) In the speculative case where working-class preference exceeded middle-class preference, an odd pattern emerged:

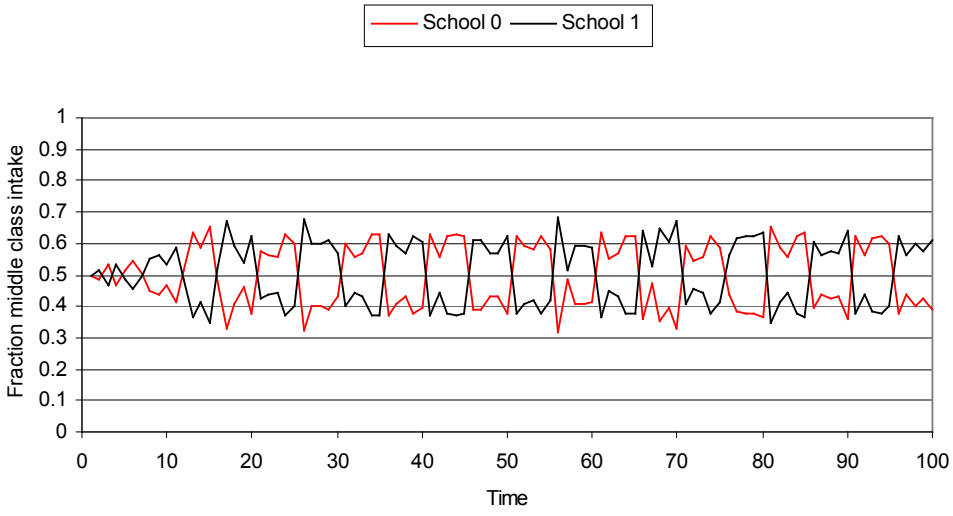


Figure 5.3 - Middle-class preference of 2.56 with a much larger working-class preference of 20.48, causing continual switching of school league table positions

The effect in c) is due to any slight increase in either school's middle-class intake making it vastly more attractive to working-class families, but only mildly more attractive to middle-class families. This school therefore experiences a boom in working-class applications; this however brings its middle-class intake down again, since any middle-class family choosing this school will be competing against an unusually large pool of

applicants. The rejected middle-class applicants therefore re-apply to the other school, joining the ones who already chose the 'second-rate' school in the first round. The schools therefore switch roles continually. The small lag between switching roles is because league table ratings are an average of the middle-class intake over the past four years, so differences in the school's composition take a little while to have a strong effect.

This effect was observed in all cases where the working-class preference exceeded the middle-class preference, provided the working-class preference is high enough to produce significant (temporary) differentiation. Since this seems logical, we will view this as further informal evidence that the model is functioning correctly. But this phenomenon does not seem to warrant further attention, since it lacks realism.

Case b), in which differentiation between schools emerges, can be validated more thoroughly with regard to Room and Britton's model. Their Mathematical Annex (Room and Britton, 2006b) includes a calculation

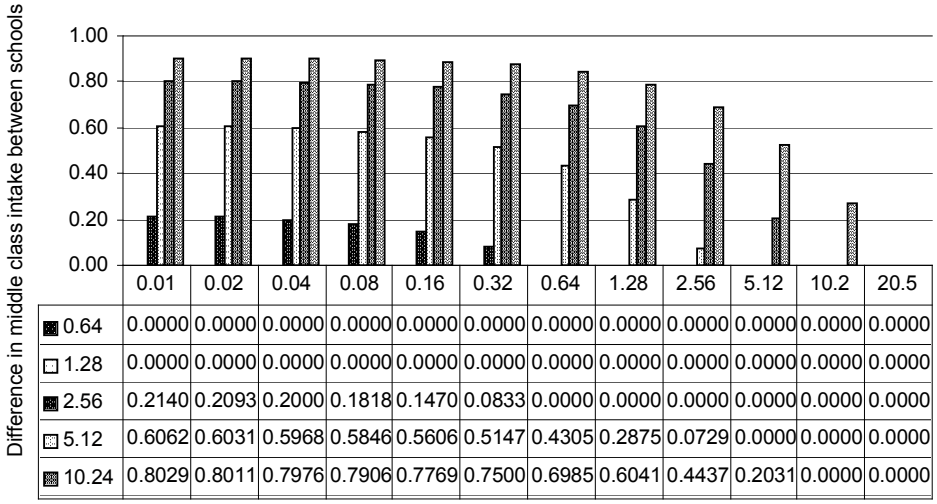
$$z^* = \frac{2\theta(1-\theta)(a-b)-1}{\theta a + (1-\theta)b},$$

Equation 5.1 - Predicted inequity z* for class blind schools

where, as before, θ is the fraction of the population that is middle-class (currently fixed at 0.5), *a* is the middle-class preference for middle-class schools, and *b* is the equivalent working-class preference. This gives the expected 'gap' or inequity between two class blind schools.

If their model does not predict any significant differentiation between schools, the formula produces a negative value, which has here been truncated to 0, to allow comparison with simulation results.

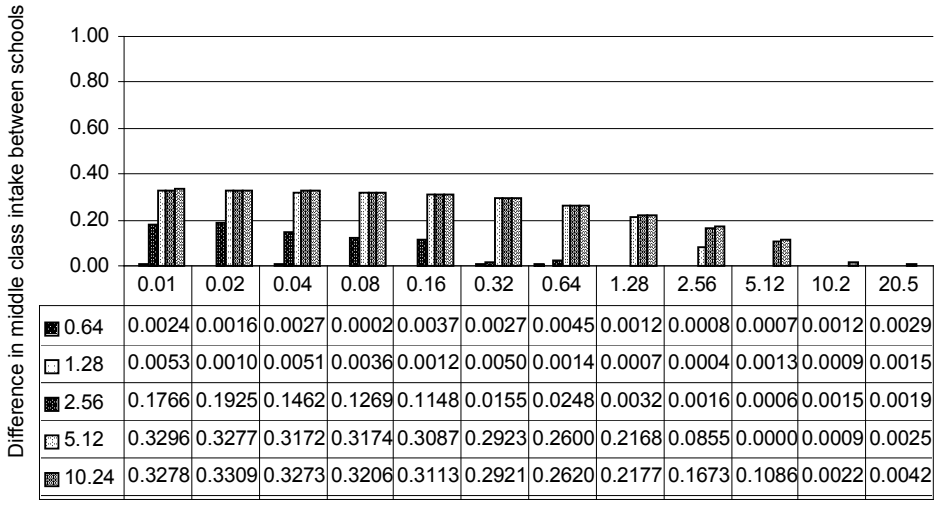
*z** was calculated for each of the simulation runs in this test, producing the following prediction:



	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.28	2.56	5.12	10.2	20.5
0.64	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1.28	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2.56	0.2140	0.2093	0.2000	0.1818	0.1470	0.0833	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5.12	0.6062	0.6031	0.5968	0.5846	0.5606	0.5147	0.4305	0.2875	0.0729	0.0000	0.0000	0.0000
10.24	0.8029	0.8011	0.7976	0.7906	0.7769	0.7500	0.6985	0.6041	0.4437	0.2031	0.0000	0.0000

Figure 5.4 - Inequity between class blind schools, as predicted by Room and Britton's model. The vertical axis shows the predicted inequity between the schools' fraction of middle-class pupils. The bars are grouped by working-class preference running along the horizontal axis (0.01-20.48), with each shade of bar representing a different middle-class preference (0.64-20.48). The actual data values are shown below to draw attention to all the 0 values.

Their model predicts no inequity will occur with middle-class preference 1.28 or lower; and if the working-class preference is ≥ middle-class preference. Next, the actual inequities obtained in the simulation runs were measured:



	0.01	0.02	0.04	0.08	0.16	0.32	0.64	1.28	2.56	5.12	10.2	20.5
0.64	0.0024	0.0016	0.0027	0.0002	0.0037	0.0027	0.0045	0.0012	0.0008	0.0007	0.0012	0.0029
1.28	0.0053	0.0010	0.0051	0.0036	0.0012	0.0050	0.0014	0.0007	0.0004	0.0013	0.0009	0.0015
2.56	0.1766	0.1925	0.1462	0.1269	0.1148	0.0155	0.0248	0.0032	0.0016	0.0006	0.0015	0.0019
5.12	0.3296	0.3277	0.3172	0.3174	0.3087	0.2923	0.2600	0.2168	0.0855	0.0000	0.0009	0.0025
10.24	0.3278	0.3309	0.3273	0.3206	0.3113	0.2921	0.2620	0.2177	0.1673	0.1086	0.0022	0.0042

Figure 5.5 - Actual inequity observed between class blind schools, following the same format as Figure 5.4.

These were much lower than expected for high middle-class preferences, but relatively accurate for low values of middle-class preference. None of the cases show a difference of exactly 0, but that is to be expected, due to the effects of randomness. Figure 5.6 shows the differences between the expected and actual values:

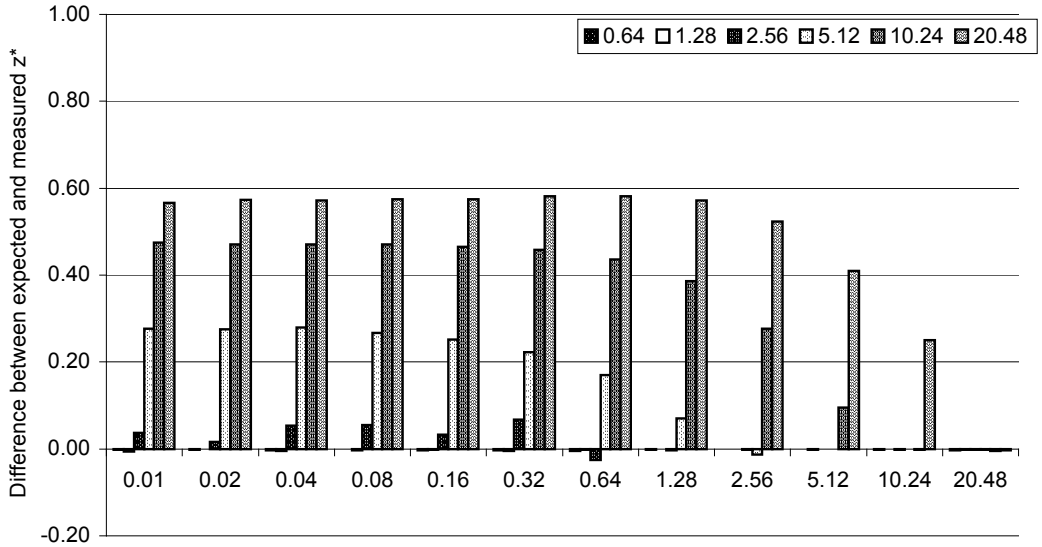


Figure 5.6 – Differences between the predicted values of Figure 5.4 and the actual values of Figure 5.5. It follows the same format as they do, so that the horizontal axis represents categories of working-class preference, while each bar shade stands for a different middle-class preference.

It is clear that our model does not correspond well to the equations' prediction with high values of middle-class preference. A second experiment takes a closer look at its correspondence with lower, less extreme middle-class preferences, ranging from 1.25 to 2.45 (and using only a subset of working-class preferences, 0.01, 0.08, 0.16, 0.32, 0.64 and 1.28):

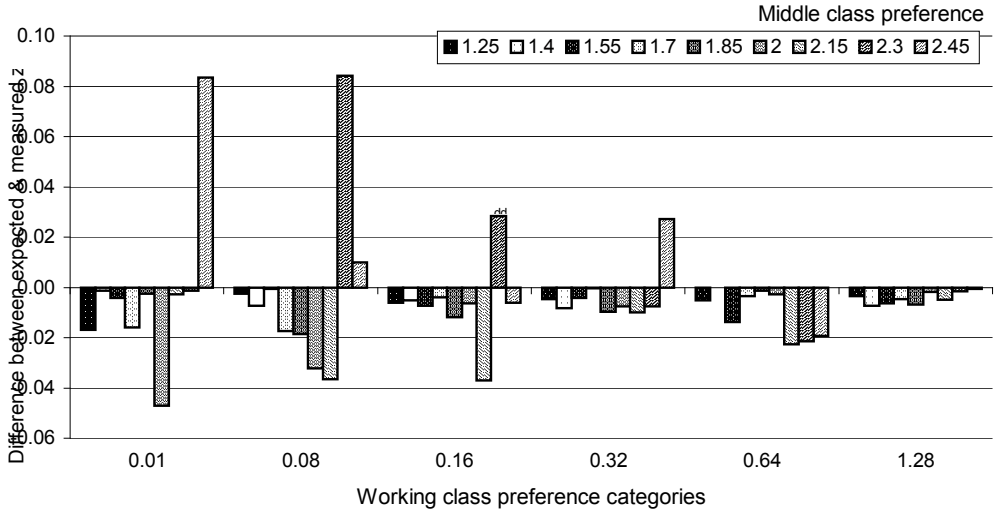


Figure 5.7 – Differences between predicated and simulated school inequity, focusing on 'normal', non-extreme middle-class preferences. The model corresponds much more closely to expectation within this range, with a mean difference magnitude of only 0.0130. (Note that the scale of this figure is 10x that of Figure 5.6).

Based on this evidence, we can conclude that the model performs according to expectation with normal values of middle-class preference, but that it does not produce as high a differentiation with extreme values as Room and Britton's model predicts. Either this model is not reliable with very high values of middle-class preference, or Room and Britton did not intend such high values to be described by their model, perhaps deeming them unrealistic.

The only restrictions Room and Britton placed on their model's middle and working-class preferences a and b are that $a >$ b and $a, b > 0$. However, it is suggested here that the only reason they placed no upper bound on the preferences was perhaps that an arbitrary value would have to be picked. For high enough preference values, their predictions become inaccurate, evidenced as follows:

Firstly, according to Room and Britton (2006b), referring to the class blind case, "The model does not predict that one school will monopolise all of the middle-class students. Rather, the size of the ultimate inequity is z*" (see Equation 5.1). Yet, for sufficiently high a and low b, we get $z^* \approx 1$, i.e. complete polarisation. E.g. a=100, b=0.01 gives z^*=0.98. This seems at odds with their intentions.

Secondly, working through the mechanics of school admissions, it seems clear that for the above example, z*=0.98 cannot be the case. Suppose we have two schools, A and B. A is in the lead, and because the middle-class preference is extreme, every single middle-class family selects A. So far all is well. However, even though working-class families have only a very slight preference for school A, they do not disfavour it either; and so approximately half of them select school A, choosing between A and B almost arbitrarily.

Since school A is class blind, it selects among its applicants completely randomly. Suppose there are 400 children in total; then school A will have exactly 200 middle-class applicants, and approximately 100 working-class applicants, with fluctuations year-on-year due to randomness. Then school A will select randomly from its pool of 300 applicants, resulting in a roughly 2/3 middle-class school population. There is no reason, apart from an unusual sequence of pseudo random numbers, that any school should ever be more than 2/3 or less than 1/3 middle-class in this scenario. In other words, $z^* = 1/3$ using a typical sequence of random numbers.

The above version of events supports the results obtained in this experiment fully. Thus, assuming no error has been made, it seems that the simulation has already been of use in exposing a hidden assumption in Room and Britton's model: that it only predicts accurately for non-extreme preferences. However, it seems useless to pick a number x based on experimentation and say it defines the boundary between extreme and non-extreme preferences, since $x + 0.01$ would probably produce little difference to x's result.

Room and Britton define $z^*=x_1^*-x_2^*$, where x_1^* and x_2^* describe the middle-class composition of the schools 1 and 2 respectively. Suppose school 1 has the advantage. Then, based on the reasoning described above, we can say that

$$x_1^* \leq \frac{2\theta N}{2\theta N + (1-\theta)N},$$

Equation 5.2 – maximal middle-class fraction a class blind school can contain, assuming N is large enough to average out local random number spikes

where N is the number of places in available in each school, i.e. x_1^* could never exceed numMiddleClassApplicants/numTotalApplicants[26]. (The higher the difference between middle and working-class preference, the bigger x_1^*.) Room and Britton already state that $x_2^*=2\theta-x_1^*$.

Room and Britton's z^* equation (Equation 5.1) could easily be adapted to accommodate this extra consideration we have discovered if it were truncated so that

$$z^* = \min\{z^*, 2\max(x_1^*) - 2\theta\},$$

Equation 5.3 - adapted definition for z* that supports extreme preferences as well as 'normal' ones

using the 'max' function to refer to the maximal value x_1^* can take on, as newly defined in Equation 5.2. The right-hand component is simply a reordering of existing equations, since

$$z^* = x_1^* - x_2^* = x_1^* - (2\theta - x_1^*) = 2x_1^* - 2\theta$$

The use of the min function to truncate z^* serves to enforce our observation that x_1^* has a maximal value[27].

From here onwards, the calculation of z* can be assumed to incorporate this truncation.

[26] 2θN corresponds to the total number of middle class children in this district, 2 * 0.5 * 200 = 200 in this case, which is the maximum number who could apply; (1-θ)N corresponds to half the total number of working class children, which is the minimum who could apply, assuming typical random numbers and the lowest working class preference possible (i.e. 0). (1-θ)N = 0.5 * 200 = 100 in this case, giving us $x_1^* \leq 2/3$ as desired.

[27] Continuing the earlier example, we have already calculated $x_1^* \leq 2/3$; now we can calculate that $z^*=2*2/3 - 2*0.5=1/3$, as desired.

5.1.2 Sweep of middle-class fraction of the population

Next, the original model's prediction that the closer the middle-class fraction of the population is to 0.5, the larger the inequity between schools z* shall be tested.

The case middle-class preference = 2.56, working-class preference = 0.01 was chosen, so that z* would be relatively high, and not fall away too rapidly as we move away from 0.5. The results are shown in Appendix A (Figure 9.1). They show an average error of 0.016; this seems reasonable, given that the values could range from 0 to 0.66. Therefore, having established that the results are relatively trustworthy, let us look at the actual inequity measured in the simulation run:

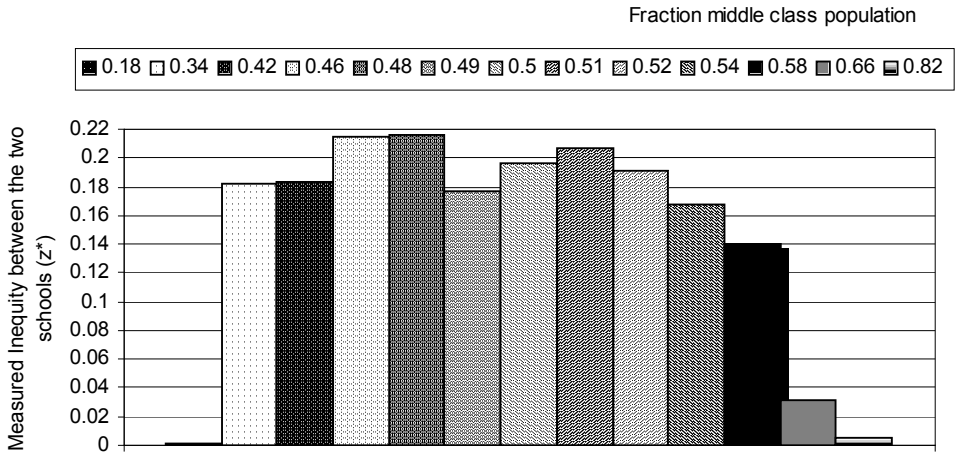


Figure 5.8 - Inequity between the two schools' fraction of middle-class intake, while varying the fraction of middle-class population. (Note that the intervals between bars are logarithmic, centred on 0.5, rather than being evenly spaced.)

Interestingly, Room and Britton's claim that the inequity will be larger the closer the fraction middle-class population is to 0.5 seems to hold for values > 0.5, but not for values < 0.5 (apart from for fraction 0.18, which exhibits no inequity). Upon examining their predictions in more detail (see Appendix A, Figure 9.2), it seems that their equations do not support their statement that a value of 0.5 will produce the greatest inequity; rather, a small peak is predicted around 0.46, which falls off to either side, but far more steeply in the < 0.5 direction. The simulation's output seems to therefore be validated by Room and Britton's more formal prediction.

The simulation was repeated with an extreme middle-class preference of 20.48 to see what would happen, and to further test our amendment to their original z^* formula. The working-class preference was kept at 0.01 to increase the 'extreme conditions'. This resulted in a very close match of the expected with the actual data, with an average error of only 0.0017. Without the adjustment to their formula suggested here, the error would have been 0.58.

5.1.3 Class blind variation found using different random seeds

For all the simulation runs so far, a constant random seed has been used, to enable fair comparison between them. However, the results have little reliability if it cannot be demonstrated that using a different random seed would not have produced a substantially different result. Therefore, three representative cases were run with 100 different random

seeds each. The simulations were run for 500 years to minimise variation due to initial conditions.

No differentiation between schools

First, the case where no differentiation between schools appears was examined. Ignoring random effects, one would expect each school to have a mean middle-class intake of 0.5 exactly, assuming the fraction of middle-class children in the population is 0.5. The more children and time steps that are used, the closer the mean should approach this value.

100 simulation runs were performed using middle-class preference 0.32 and working-class preference 0.01; the mean middle-class intake ranged from 0.496 to 0.504, i.e. it was very closely centered on 0.5 as expected. The standard deviation between the twenty schools' mean middle-class intakes was very low at 0.0008, further supporting the conclusion that the runs all followed the same essential pattern.

Having established that the schools all followed the same broad pattern, we turn towards local comparisons. As Figure 5.1 illustrated, local spikes can be relatively high, even though the intake averages out at roughly 0.5 overall.

The minimum middle-class intake during any single year in any of the 200 schools was 0.385. The standard deviation between each school's overall minimum middle-class intake was still very low, at 0.0096. Naturally, the statistics for the maximum intake in any year mirror this.

Mild differentiation between schools

Next, the case where differentiation between schools is found was considered, using middle-class preference 2.56 and working-class preference 0.08.

The schools can no longer all be compared against each other; instead, each top school should only be compared to the other nine top schools, and the same for each of the ten lower-rated schools.

Here, more variation was to be found, with the smallest minimum middle-class intake during any one year among the ten bottom schools being 0.29, and the highest minimum intake being 0.35. This resulted in a larger standard deviation than before between these minima of 0.012, and similarly for the local maxima. While these standard deviations are higher than in the previous case, the spread between results that they imply is still acceptably low to allow us to pick any one of these ten runs and call it a 'representative sample', as we have been doing so far. Furthermore, the standard deviation between the top schools' mean intakes was also low at 0.011, indicating that they differ only in local spikes. (The standard deviation for the bottom schools' intakes was identical, since each top school is the mirror image of each bottom school – see Figure 3.1.)

Pronounced differentiation between schools

Finally, the 'extreme' case having a middle-class preference of 20.48 and a working-class preference of 0.01 was considered. This resulted in a 0.0099 standard deviation between the local minima and maxima of each school type (leading or 'second-rate'), and a standard deviation of 0.0013 between their respective mean middle-class intakes. This exceptionally low spread between the results of different runs is probably because the middle-class preference is so strong that every middle-class child applies to the leading school, whereas in the less pronounced cases the number of middle-class children applying year on year fluctuates slightly due to randomness.

5.2 Class sensitive case

We now consider the second of the two cases addressed by Room and Britton, in which schools unconditionally choose middle-class applicants over working-class applicants.

5.2.1 Sweep of the middle-class preference for middle-class schools parameter

This preference parameter was clearly identified by Room and Britton as the decisive one in the class sensitive case. If sufficiently high, complete polarisation is expected to emerge between the two schools, with one accepting every single middle-class child, leaving the other with none. Stronger preferences are expected to lead towards this inequity state sooner.

Room and Britton clearly identify the exact conditions under which this polarisation is expected to occur, centred on the critical value of $a\theta = ½$ (as before, a refers to the middle-class preference for middle-class schools, and θ is the fraction of the population that is middle-class). If $a\theta < ½$, they predict that any inequity between schools will reduce over time (see experiment 6.4.10); if $a\theta > ½$, it will grow over time, (the larger a is, the faster the growth).

As discussed in section 4.2, working-class preference will be set to ½ of the middle-class preference for middle-class schools in these tests. In this experiment, we shall also fix $\theta = 0.5$ as usual. This means that the critical value cut-off point $a\theta = ½$ occurs at $a = 1$. We shall therefore sweep both logarithmically increasing and logarithmically decreasing, starting at a = 1. This resulted in the sequence $a = 0.36, 0.63, 0.84, 0.92, 0.96, 0.98, 0.99, 1, 1.01, 1.02, 1.04, 1.08, 1.16, 1.32, 1.64, 2.28, 3.56, 6.12$ and 11.24 being tested.

The number of years the simulation was run for needed to be increased from 500 to 1500, since patterns took much longer to emerge than in the class blind case.

The results reflected the prediction very precisely: polarisation was only encountered from $a = 1$ onwards; $a = 0.99$ was not sufficient. The time taken to reach total polarisation (and stay there) followed a pattern of exponential decay with increasing middle-class preference. Each time step was also simulated in Excel according to the formulae of Room and Britton's model, and the results mapped against those of our simulation:

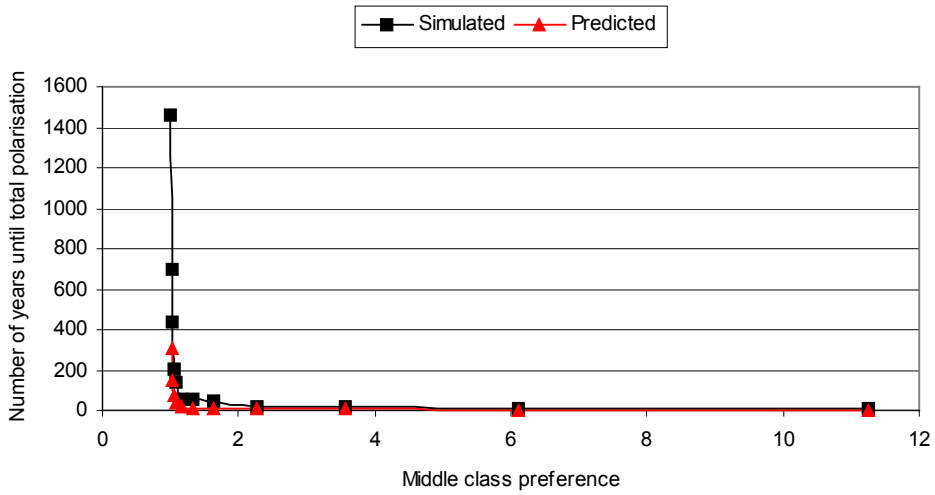


Figure 5.9 - Exponential decay of the number of years taken for complete class polarisation to be reached, as the middle-class preference for middle-class schools increases. Note that the predicted data series lacks the first (top) point in the simulated data series, because it predicts no polarisation for $a = 1$.

These calculations clarified the fact that $a = 1$ is in fact not supposed to produce polarisation, only $a > 1$, so the simulation's output was not completely accurate after all. More seriously, in every case the simulation took longer than predicted to reach

polarisation, although the overall pattern of exponential decay matches (multiplying the predicted values by e.g. 2.7 produces a much closer correspondence). Examining an individual simulation run provides some insight into why this difference occurs:

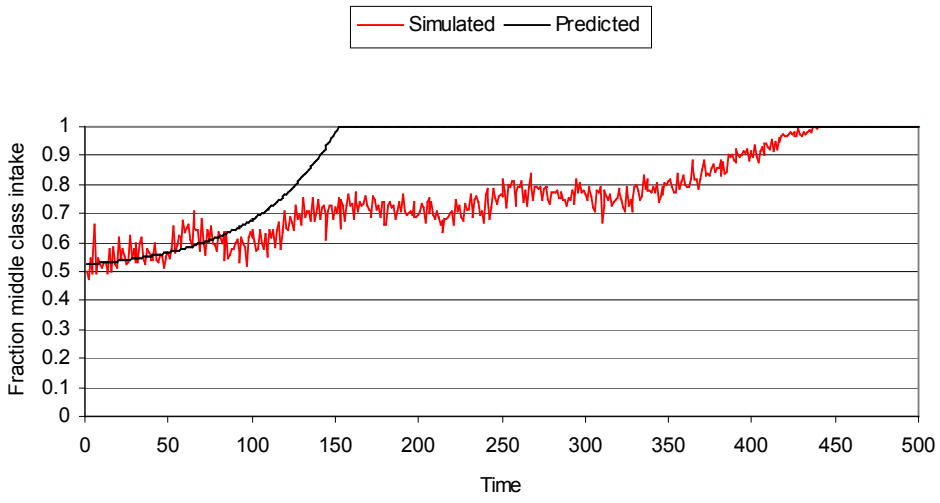


Figure 5.10 - The path predicted by Room and Britton's model for middle-class preference 1.02, and the path taken during the simulation run.

Stochastic effects appear to produce local instabilities where the trajectory temporarily proceeds downwards. This was to be expected to some extent, but it was not expected by the author that would occur to this degree. However, the higher the middle-class preference, the smaller the difference between the predicted and the simulated paths, because there is less room for random effects in a 95% likelihood than in a 70% likelihood.

With a sufficiently low preference (e.g. 0.84), the output resembles that of an undifferentiated class blind scenario very closely. The borderline case preference of 0.99 exhibits an interesting effect:

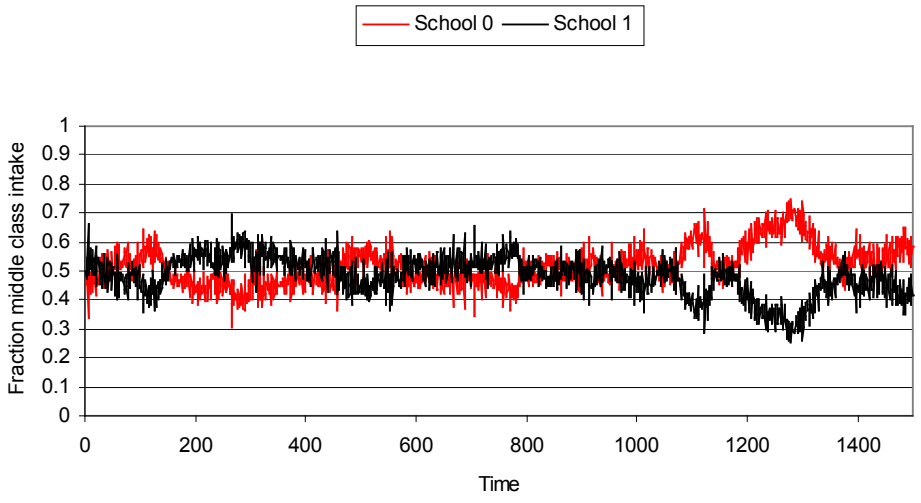


Figure 5.11 - Middle-class preference of 0.99 is not quite sufficient to produce total polarisation.

The preference is not quite high enough to produce total polarisation: local peaks can be observed, but no stable state can be reached. The state predicted by Room and Britton's model is a stable 0.5/0.5 equality; this would in theory be equivalent to the class blind case, but we are seeing more extensive local peaks than we do in the class blind scenario.

If the middle-class preference is increased to the critical value of 1, polarisation occurs, but it takes an extremely long time, and follows some failed, unstable attempts:

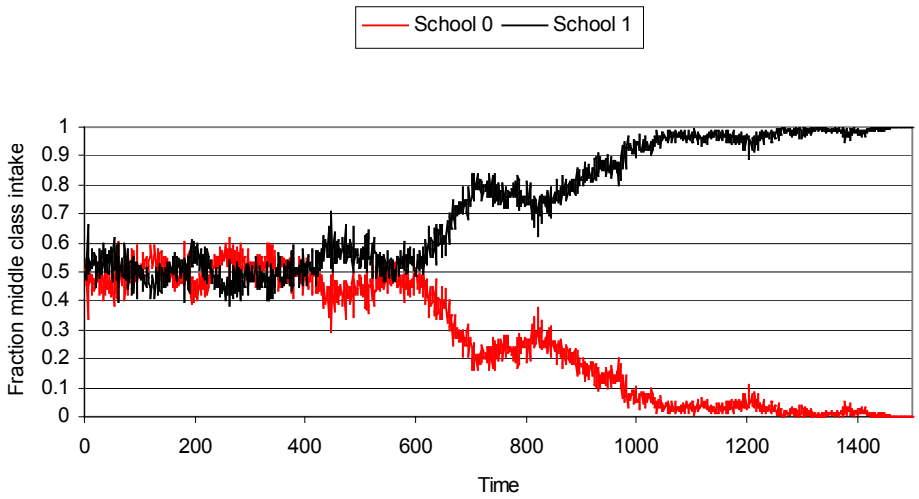


Figure 5.12 - Polarisation emerging with the critical value of middle-class preference = 1.

In fact, if the simulation is only run for, say, 500 years, it appears as though polarisation is impossible, as Room and Britton's model suggests.

In contrast, a higher preference produces much more rapid polarisation, as predicted (see Appendix A, Figure 9.3). The highest middle-class preference, 11.24, produced total polarisation in the fifth year. The minimum number of years it could take is intuitively three: in the first year, the schools are initialised with no differentiation. Consequently, in the second year, everyone chooses randomly; in the third year, every single middle-class family chooses whichever school emerged as a slight leader in the previous year, due to the Matthew effect described by Room and Britton (2006b). In fact, it could conceivably take only two years, if coincidentally every middle-class family picked the same school in the second year, but that would only be an artefact of randomness which would not occur in real life, which is made less likely the more children are used.

In summary, the test of varying the middle-class parameter in the class sensitive case was relatively successful; the results all followed the pattern predicted by Room and Britton. The unpredictable and variable nature of the results is probably inherent to stochastic simulations, and it could be argued to approach reality more closely than the equations. In real life, patterns are not mathematically perfect, especially where choices made by such complex systems as human beings are involved. For example, a family might wish their child to attend a school close to another relative's house for childcare arrangements.

5.2.2 Determining variance due to different random seeds

Especially the case $a = 0.99$ is intuitively suspicious in the previous experiment, in that one suspects that the potential for polarisation might yet be there, were it only to run a little longer.

The experiment was therefore re-run under two conditions:

a) Giving it more time – yet even after 5000 years, no polarisation occurred. Nor was either school able to retain a leading role permanently: they switched roles eight times.

b) Using 100 different random seeds, over 500 years. None of these additional runs exhibited polarisation – however, some approached it more so than others. In some cases, on average, a clear inequity between the two schools was present (0.28 in the most severe case); in others, the average middle-class intake fraction was 0.5 for both schools. The average inequity among the 100 runs was 0.11, with a standard deviation of 0.07. Examining the individual cases in more detail reveals that the cases with little average inequity tend to have several smaller 'pockets' of inequity instead, with the leading school varying from pocket to pocket, thereby cancelling out the local inequities. However, all runs had a similar maximum height of local spikes in the middle-class intake of a school, ranging from 0.64 to 0.70, indicating that the same local patterns were followed in each case.

Despite the variability in inequity, all 100 runs produced the same overall result of no polarisation occurring. While Room and Britton's model does not predict polarisation in this case, it does not say that inequity cannot nevertheless occur to a less marked degree; and indeed this intuitively seems to be logical. It therefore seems that the simulation is performing to expectation in exhibiting this volatile inequity under these borderline conditions.

A middle-class preference of 3.56 was also chosen to be rerun 100 times with different random seeds, to see how sensitive the amount of time taken to achieve polarisation was to initial conditions. On average, it took 12 steps to reach polarisation, with a standard deviation of 3. This shows that the variability is still present even in a non-borderline case. It is, however, much more prevalent near the critical value of 1.0, as is to be expected. Runs using a middle-class preference of 1.01 were also rerun 100 times; in seven of the cases, polarisation was not even reached within 1500 years. In the remaining cases, it was reached on average after 820 years, but with a huge standard deviation of 276 years. Results ranged from 369 to 1449 years. This indicates that for these borderline cases, there simply is no 'representative run'.

5.2.3 Verify that working-class parental preference has no significant impact

Room and Britton predict that the working-class preference is irrelevant in determining the eventual outcome of class sensitive school admissions. This makes intuitive sense because a middle-class child is essentially guaranteed a place at its first choice school, unless the school has more middle-class applicants than there are places. The working-class children simply fill up whatever spaces remain, regardless of their choices.

The difficulty in this test lies in the fact that varying the working-class parental preference is nevertheless expected to have a basic effect of changing which random numbers are used for what purpose in the simulation (using the same stream of pseudo-random numbers), which will inevitably introduce a certain amount of random noise variation. However, it should still be possible to ascertain whether the overall pattern is the same, in the same way that runs of the same parameters using different random seeds were compared in experiment 5.2.2.

The range of interesting values of a (the middle-class preference) was identified previously as ranging from 1.0 upwards, and so the representative parameters 0.84, 0.99, 1, 1.64 and 11.24 were chosen. Regardless of which working-class parameters they are paired with, the same pattern as found in experiment 5.2.1 should be found. This ranges from conditions where no polarisation is expected, over the boundary condition of the first value of a in which polarisation is tentatively expected, to values of a in which polarisation is most certainly expected. In each case, the usual logarithmically increasing

working-class preferences were used, from 0.01 to 10.24 (including cases where it exceeded the middle-class preference).

As expected, much more variation was found in the borderline cases 0.99 and 1 than in the others. When a = 0.84, none of the eleven working-class preferences caused polarisation to emerge, as predicted for a value of a that lies well below the critical value. In each run, the schools' average fraction of middle-class intake was clustered around 0.5 with a low standard deviation of 0.0030, showing that little differentiation emerged. From these statistics alone, one could easily imagine a repeat of the class blind case in which only very brief, local spikes were seen, but in fact the spikes here could last around 100 years. However, the inequity is averaged out over the 1500 years we examined.

When a = 0.99, again no polarisation was found, as predicted. However, both the inequity between schools and the variability in trajectories between runs was much more volatile than in the previous case. In all but two of the eleven runs, a marked inequity was observed between the two schools on average (although still subject to switches in roles, as before). The mean inequity was 0.12, very close to the 0.11 observed from 100 different random seeds earlier, although twice the standard deviation was found (perhaps excused by the significantly smaller sample used). This could give us some cause for concern that the working-class preference has an effect on the outcome after all, but there is no discernible relationship between it and the mean inequity (see Appendix A, Figure 9.4).

Therefore, even though somewhat more variability was seen between runs than when merely varying the random seed, we can still say with confidence that the test passes, especially since this case covers a borderline test where a lot of variability is expected.

Next, the case a = 1 was investigated. This case had already caused problems earlier when it exhibited polarisation after an extremely long time, even though none was predicted, lying as it does precisely on the boundary between polarisation and non-polarisation. In two of the eleven cases, total polarisation occurred; however, another showed signs that it could have reached polarisation, given more time, and the same may hold for the others. One run was particularly interesting:

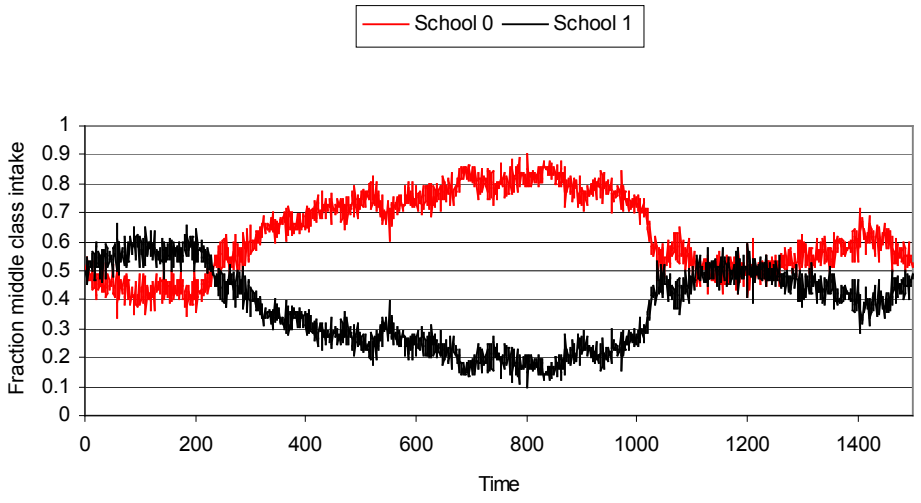


Figure 5.13 Interesting run of middle-class preference 1 with working-class preference 0.08, where it appears there is a tendency towards total polarisation, but this completely subsides again.

It illustrates nicely that it can appear as though total polarisation is being worked towards, but in this borderline case, a stable polarisation is not assured until it has been fully reached.

The next test involved $a = 1.64$, in which polarisation is definitely predicted to occur. The dependent variable here is the number of years this takes. This is again represented as a scatterplot in Figure 9.5, Appendix A. As before, there is no discernible relationship, and so we can conclude that the working-class preference plays no role in the outcome of the simulation. However, clearly the variability between runs was still relatively high: the mean number of years was 40.5, with a standard deviation of 9.6. By way of comparison, the working-class preference 0.01 case was chosen to be re-run with 11 different random seeds, producing a mean of 32.4 years with a standard deviation of 11. Taking into account the small sample size, it seems that the variability obtained from varying the random seed and that obtained from varying the working-class is approximately the same, i.e. the working-class preference has no significant impact.

Finally, the extreme case $a = 11.24$ was run. Being far from the borderline case, very little variance was exhibited, as expected. In fact, polarisation occurred in the same year, year 5, for every working-class preference used. As one might expect, the trajectory taken still differed between runs, but only very weakly: studying the first five years only, a low standard deviation of 0.0049 was found between the mean intake of the leading schools, based around a mean of 0.691.

In summary, the experiments run indicated that the working-class preference does indeed have no significant impact on the simulation's results. However, repeating more of the experiments, and widening the parameter space investigated, would lend greater confidence to the finding.

5.2.4 Varying the fraction of the population that is middle-class

For the class sensitive scenario, Room and Britton make no prediction about a specific value of θ (the fraction of the population that is middle-class), as they did for the class blind scenario; rather, their predictions regarding θ are embodied in the concept of the 'critical value' of $a\theta = ½$ at which polarisation begins to occur. We previously investigated varying only a (the middle-class preference) in experiment 5.2.1; we now vary θ for certain representative values of a. For $\theta = 0.5$, these representative values were chosen as 0.84, 0.99, 1, 1.64 and 11.24 since the critical value was 1; more generally, we now use $1/(2\theta) - 0.16$, $1/(2\theta) - 0.01$, $1/(2\theta)$, $1/(2\theta) + 0.64$ and $1/(2\theta) + 10.24$ as our representative values of a.

Having demonstrated that the working-class preference has no significant impact on results, it was set at 0.01 for simplicity[28]. Since no special importance is attached to any particular value of θ, it was varied linearly from 0.1 to 0.9 in increments of 0.1.

[28] Calculating it at ½ the middle class preference requires multiple parameter file specifications to be used for a single experiment, since Repast does not support the description of one parameter in terms of another (unless a custom parameter file reader were to be written). See section 8.3.6.

Theta	1/(2*theta) - 0.16		1/(2*theta) - 0.01		1/(2*theta)		1/(2*theta) + 0.64		1/(2*theta) + 10.24	
	Simulated	Predicted	Simulated	Predicted	Simulated	Predicted	Simulated	Predicted	Simulated	Predicted
0.1	/	/	/	/	1038	/	105	27	19	5
0.2	/	/	/	/	1358	/	77	16	7	4
0.3	/	/	/	/	527	/	92	10	5	4
0.4	/	/	/	/	unstable	/	30	13	10	4
0.5	/	/	/	/	/	/	43	9	5	4
0.6	/	/	/	/	unstable	/	23	7	4	3
0.7	/	/	/	/	/	/	15	5	3	3
0.8	/	/	/	/	unstable	/	15	4	3	3
0.9	/	/	/	/	unstable	/	9 (blips)	3	3	3

Table 5.1 – Years taken for total polarisation to be reached. Rows show values of θ (middle-class fraction of population) being varied; the double-columns represent increasing values of middle-class preference, calculated from θ. The left-hand side of each double-column gives the simulated number of years, while the right gives the number predicted by Room and Britton's model. 'Unstable' means that polarisation was approximately reached, perhaps with many little jittery blips; and that it was then left again. 'Polarisation' was defined as four or more successive years where the leading school has a 100% middle-class intake, since a league table spans four years in the simulation.

It should first be noted that the prediction based on Room and Britton's equations never predicts an unstable state. When initialised with the random choices made by families in year 1, it either decreases steadily towards a fixpoint of precisely θ if no polarisation is predicted, or it increases steadily towards a fixpoint of precisely 2θ (or towards 1 if θ > 0.5, since of course a school cannot contain more than 100% middle-class pupils).

Examining the case of θ = 0.9, middle-class preference = 1/(2θ) + 0.64, shown below,

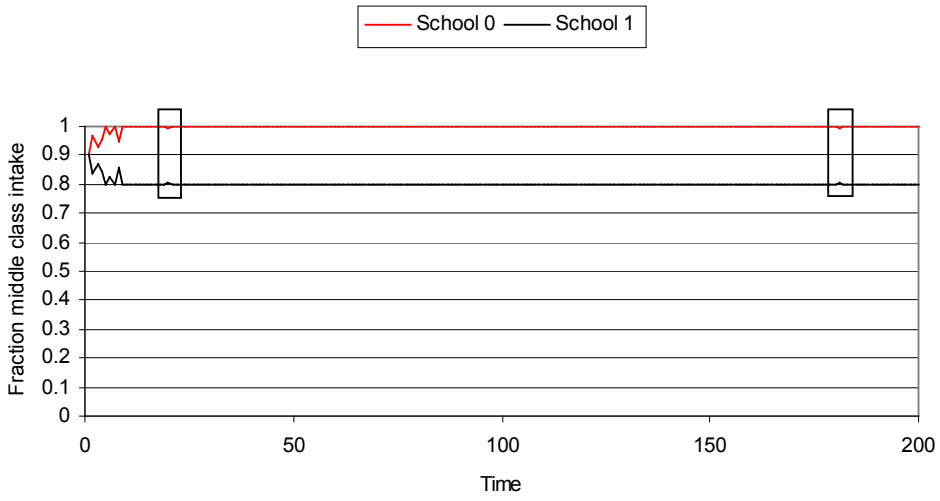


Figure 5.14 – Polarisation when fraction middle-class = 0.9, middle-class preference = 1/(2θ) + 0.64 = 1.2. The 'blips' mentioned in Table 5.1 are enclosed by rectangles. This is a previously unseen phenomenon.

it is clear why, even with such a high middle-class preference, having such a large value of θ means that the schools are not differentiated enough to prevent the occasional 'blip' where the leading school cannot quite achieve a 100% middle-class intake. It is because even when the school has a full league table history of a 100% middle-class intake, the other school is nevertheless also an excellent candidate with a 90% middle-class intake.

Thus, with the given middle-class preference, only a 62% probability can be allocated to families preferring the leading school. In the mathematical model, this is more than enough to produce total polarisation – exactly 62% of the 360 middle-class pupils will apply to the leading school (assuming 400 children in total as usual), which results in 223 middle-class applications; plenty to fill its 200 places. However, in the simulation, theoretically every single family could happen to fall into the 38% likelihood category that the other school will be chosen, since their choices are independent of other families' choices in that year, unlike in the macro model. Of course that is extremely unlikely[29]; what is more probable, and indeed what occurred in the simulation run, was that occasionally less than 200 middle-class parents would choose the leading school, resulting in the so-called 'blips'. If only 199 applications were made to the leading school in a given year, then the 62% probability would have been played out as 55%, which is not unreasonable as an occasional occurrence. Experiment 5.3.1 shows how using a larger number of children results in less such random noise.

It is unsurprising that the greatest discrepancies between the simulated and the predicted results occurred during the borderline case. Unfortunately, apart from the random fluctuations already explained, the simulations are also subject to floating point inaccuracies. Room and Britton's predictions themselves show how sensitive the outcome is to the exact value used; for $\theta = 0.9$, $1/(2\theta) = 5/9$ which does not predict any polarisation since leads directly to a fixpoint, but if this is rounded to a finite approximation, polarisation is mistakenly predicted (the more accurate the approximation, the slower the move towards polarisation).

The remaining cases were as expected, given that we have already established that the simulation takes a proportionally longer time to reach polarisation than the mathematical model. It can therefore be concluded that the simulation operates as well with varying values of θ as it does for $\theta = 0.5$.

5.3 Confounding variables

5.3.1 How many children are used

Finally, an experiment was undertaken to confirm that the number of children used in the simulation was not too low to act as a representative sample, and therefore acting as a confounding variable by distorting the results.

An average class blind case of middle-class preference 5.12, working-class preference 1.28 was chosen, since the difference when varying the number of children was most obvious in this type of case, where a clear inequity, but not total polarisation emerges.

Usually, 200 children per school were used in the experiments. We tested the effect of using 10, 100, 500, 1000, 2000 and 5000 children per school instead. 10 children were clearly too few, and no stable state could be achieved. The run using 100 children did experience differentiation between schools, but this was lower than the other runs', at only 0.18 compared to their mean inequity of 0.21. Among the other runs, the variability among their mean inequities was very low, with a standard deviation of only 0.00092.

Nonetheless, using more children did make a difference:

[29] The statistical likelihood is 0.38^{360}, where 1 is absolute certainty and 0 represents an impossibility.

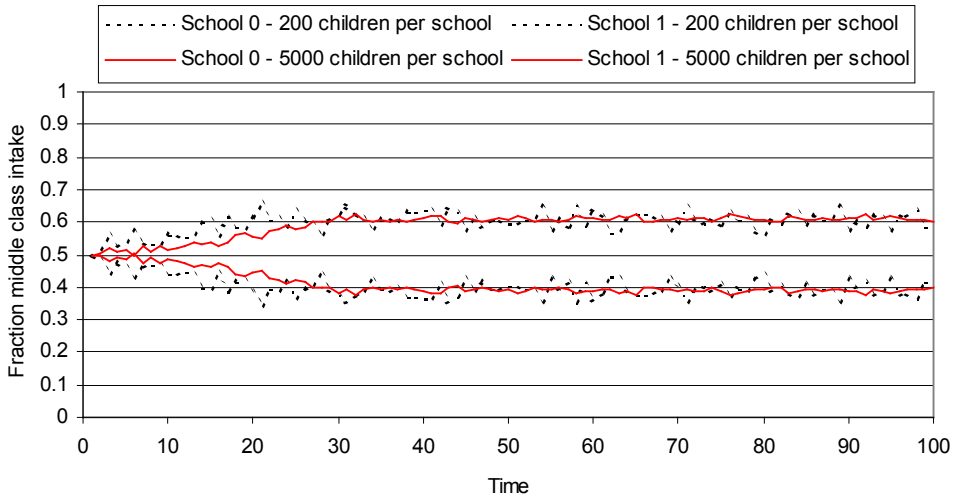


Figure 5.15 – Two class blind runs are superimposed here; in dotted black, the original 200 school places run, and in red, the same experiment using 5000 places per school. The middle-class preference is 5.12, and the working-class preference is 1.28.

As seen above, using more children smoothes the schools' trajectories noticeably. This is because a larger sample size causes statistical abnormalities to be averaged out to a greater degree. Room and Britton's mathematical model predicted a completely static inequity from one year to the next; the simulation should in theory also attain this result if an infinite number of children could be used. However, the local spikes caused by using a low sample size of only 200 children per school is in fact a benefit of simulation, not a flaw. Real schools also only have a similarly 'small' capacity, and consequently the statistical abnormalities seen in our simulation no doubt also occur in the real world, which is not as statistically perfect as an equation. Thus, the model approaches the dynamics of real life trajectories more closely than the mathematical model, which takes a much more global view.

5.3.2 Using an odd number of children

It was postulated that using an odd number of children could potentially cause problems, since one school would have an empty place every year (if 401 children are used, each school is allocated 201 places). However, this was not found to be the case: when schools were not significantly differentiated, the empty place sometimes lay in one school, and sometimes in the other; otherwise, the less popular school would always have the lesser number of applicants. Having 201 middle-class applicants against 200 working-class applicants did not noticeably affect the results either.

6 Experimental Results: Extensions

Having validated the basic Java prototype, and found it to correspond acceptably well to Room and Britton's model, the prototype was extended using evolutionary development, to allow various scenarios of interest to be researched. Since these additions cannot be validated against the original model, they must instead be validated against experimental predictions where possible, and examined empirically in their low-level agent behaviour to see if the underlying processes are as expected. However, this reduced level of validity remains; and for the simulation to be viewed as more reliable, avenues such as comparisons against real field data should be explored.

Stochastic shocks, using many schools and the introduction of children's preference for class blind schools produced useful results, but the school specialisms did not appear to have a significant effect, on the whole.

6.1 Stochastic shocks

The first extension to be explored was the possibility of a 'stochastic shock' being applied to one or both schools, representing, say, a favourable news report. Questions of interest were: how strong such an effect would need to be to cause differentiation to emerge between schools where it would otherwise not appear; and how much differentiation could be present between schools before the stochastic shock would fail to switch their roles.

6.1.1 Non-binary school strategies

Class sensitive schools were not considered, because it would be exceptionally rare in real life to encounter a sufficiently stochastic shock to switch the 100%/0% middle-class roles of the schools. Instead, scenarios with only partial differentiation between schools were used. Rather than controlling the degree of inequity through the difference between the middle- and working-class preferences as previously in the class blind case, this was achieved through the introduction of a preference schools have for middle-class children. This replaced the previous binary class blind/class sensitive school strategies. A value of 1 means that the school prefers middle-class children no more than working-class children, i.e. that it is class blind; a value of $x > 1$ means it prefers them x times as much. The maximum value Java allows for an integer (2147483647) was found to replicate the class sensitive behaviour exactly. Figure 9.6 in Appendix A shows the resultant exponential increase in inequity with increasing school preference for middle-class applicants.

6.1.2 Stochastic shocks triggering differentiation

Four cases were investigated to see whether a stochastic shock could trigger differentiation between schools, or exacerbate it. These were:

a) No intrinsic inequity (school preference for middle-class of 1.08)

b) Minimal, unstable intrinsic inequity (school preference of 1.16)

c) Minimal but stable intrinsic inequity (school preference of 1.32)

d) Clear intrinsic inequity of around 0.16 (school preference of 1.64)

Each of these cases was tested with a positive stochastic shock to the bottom school during a 500-year run, with the values 0.5, 0.4, 0.3, 0.2, 0.1 and 0.05[30]. (This is the temporary 'bonus' to the school's league table rating; its effect decays over time.) Negative shocks to the leading school (representing e.g. bad press) were not considered in detail, since they were found to just be the mirror image of a positive shock to the bottom school.

The results showed that a stochastic shock is not able to produce differentiation where none is naturally possible: none of the stochastic shocks were able to effect more than an almost undetectable increase in the affected school's rating in case a). Even when the shock was increased to the unrealistic maximum possible value of 1, nothing happened.

In case b), all shocks except 0.05 succeeded in switching the schools' roles, but only temporarily; this must be due to the intrinsic instability of the system. In case c), again all shocks except the smallest succeeded in reversing the schools' roles, but this time the resultant state was stable. Case d) is identical except that only shocks ≥ 0.3 are able to overcome the significant existing inequity. The crucial finding for experiments b, c and d was that the stochastic shock is at most able to produce a change in the schools' roles; it cannot alter the degree of inequity between the schools. The inequity appears to be built in to the system defined by these preference parameters. One might have imagined that in a 'weak differentiation' case, given a sufficient head start, a school might be able to cling to the artificial advantage it was given; but this is evidently not the case.

6.1.3 Stochastic shocks switching the roles of schools

The second experiment undertaken on stochastic shocks was to determine the combinations of shock strength and existing inequity (set by the school preference parameter) under which the switch of school roles already seen in the previous experiment takes place. If the shock is too weak, or the inequity is too high, the transition will not occur.

The following values of stochastic shock were investigated: 0.3, 0.2, 0.1, 0.025, 0.005, 0.001, and 0.0005. Values > 0.3 were not considered of interest since it is hard to imagine any realistic event having such a profound effect on a school's reputation.

Each of these stochastic shocks was applied to scenarios with school preferences for middle-class applicants of 1.0 to 2.6, in increments of 0.05. The upper limit of 2.6 was decided upon through informal experimentation: no change was observed for higher values. The experiment was initially performed over 500 years, but having noted that it makes no difference, subsequent re-runs used 200 years instead.

The results were as expected: higher stochastic shocks were more likely to cause the schools to switch roles, as was a lower degree of inequity. However, the inequity also had to be high enough to allow differentiation to emerge between schools, so no effect was seen for school preferences from 1.0 to 1.10.

The range of school preferences in which a stochastic shock of 0.3 caused the role reversal was 1.15 to 2.15, with an 'uncertain zone' from roughly 2.20 to 2.40, since the role reversal occurred unexpectedly at 2.25 and again at 2.35. The reason for this can be seen in the diagram for 2.25:

[30] Good fortune meant that our usual random seed used in all the experiments happened to hit only the bottom school, in the middle of the run – the perfect scenario, but not guaranteed since which school(s) are affected, and when, is random – so we did not need to compromise comparability between experiments by using a different random seed (see section 8.3.6).

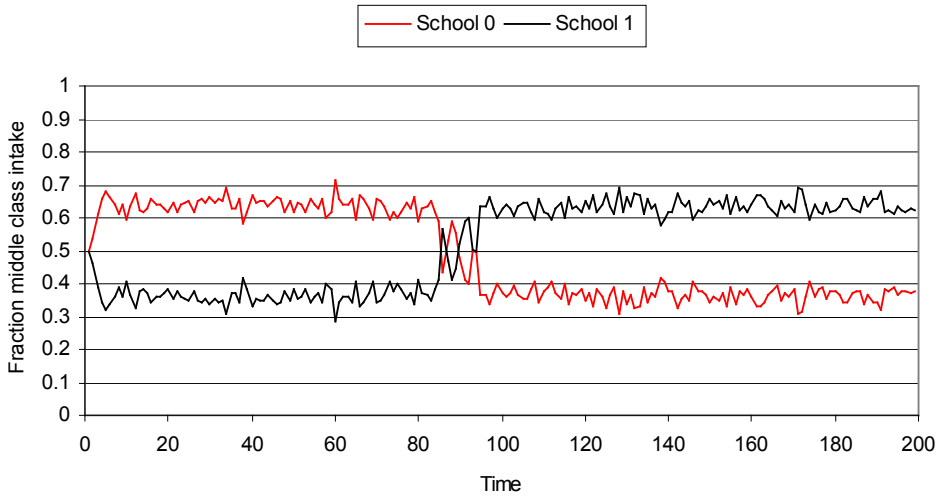


Figure 6.1 – Stochastic shock of 0.3 applied to school 1 during year 84, when the schools' preference for middle-class applicants is 2.25.

Unlike in the simulations with lower school preferences (i.e. lower inequity between the schools), the shock does not cause an instant role reversal: there is a period where it could go either way. At the top end of school preferences, it is therefore impossible to predict whether this size of shock will cause a role reversal or not.

A less powerful shock of 0.2 has a smaller 'sensitive zone' of role reversals when school preferences range from 1.15 to 1.55, with an 'uncertain zone' where the reversal might occur from 1.6 to 2.15, approximately. However, the upper limit of this zone is difficult to identify, because unexpectedly, we cannot even predict with certainty whether a local spike will occur or not. For example, the school preference of 2 aberrantly produces such a spike (similar to that of Figure 6.1, but with each school returning to its original position), despite the shock not having any perceptible impact on the run using a preference of 1.9. The lack of local spikes in those few instances such as 1.9 seemed suspicious, and so the agents' behaviour was examined more closely in those cases. This showed that the numerous cases in which either no spike, or only a minor one occurs, are in fact due to slight statistical abnormalities. Even though the bottom school should suddenly have been preferred more than previously, the random choices made by applicants happened not to turn out that way in the year of the stochastic shock. Since the stochastic shock's effect wears off over time, if it does not gain momentum in the year in which it occurs, it is even less likely to do so in the following year. Such abnormalities are more likely to occur when the increase in preference is slighter; this is why the phenomenon was not observed for the larger stochastic shock of 0.3.

The next smallest shock, 0.1, again exhibited similar behaviour, with the school switch occurring for preferences of 1.15 to 1.25, as well as 1.35. The same explanation as above applies for the lack of a rigid cut-off point.

The subsequent shock of 0.05 was not able to achieve a role reversal; all it accomplishes is temporary confusion, as shown below:

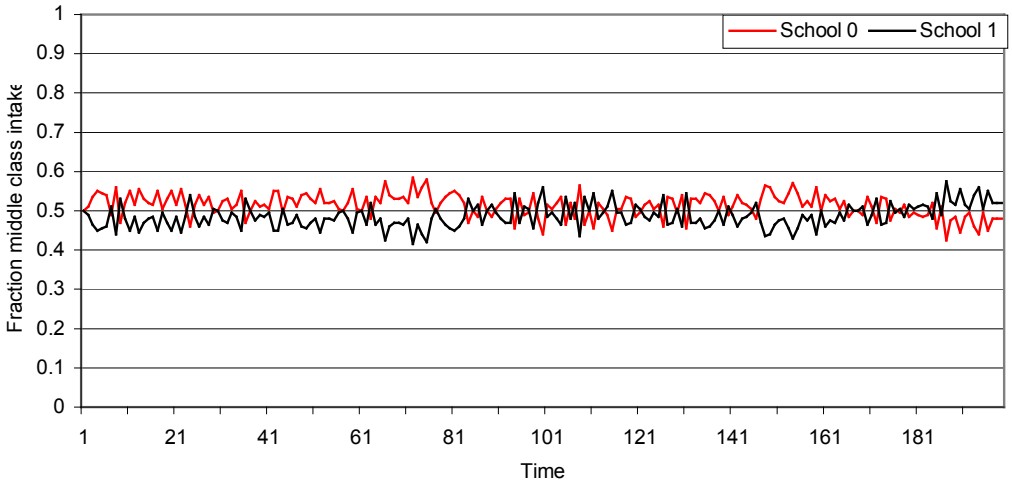


Figure 6.2 – Stochastic shock of 0.05 applied to school 1 in year 84 only causes local confusion; it cannot switch the schools' roles permanently. Coincidentally, school 1 later emerges as the leading school around year 180 anyway, due to the instability of the system.

However, the next two shocks of 0.025 and 0.005 are able to switch the schools' roles again after all (although only for a school preference of 1.15), so some additional experiments were run using even smaller stochastic shocks to establish how large a shock must be to have an effect. Shocks of 0.000 5, 0.000 05 and 0.000 005 were still able to effect a role reversal, albeit only a temporary one, due to the unstable nature of the system when the school preference is so low. However, shocks of 0.000 000 5, 0.000 000 05 and 0.000 000 005 could not achieve a lasting effect, although surprisingly they were still able to cause a small surge. These small surges were, however, smaller than others, routinely seen, that are due simply to randomness in this scenario.

In summary, it is possible to identify zones of interest for a given strength of stochastic shock where, based on schools' strategies (preferences), they are either immune to its effect, or in a 'danger zone' of being affected by it. Rather than taking a school preference scenario and seeing what shocks might affect it, we have taken a given shock and seen what scenarios it can affect, since the school preference is potentially under policy-makers' control, but stochastic shocks and their strength are in general not. This would allow an analysis to take a potential risk such as a high-profile crime at a school, and see under which school-level policies the stability of the system would remain unaffected.

The analysis could easily be reversed to allow policy-makers to plan their own 'stochastic shock', e.g. a campaign spotlighting the positive qualities of a failing school, to tell them how powerful an effect they would need to generate in order to upset the balance of the system. Of course, this would only be of use when coupled with other strategies designed to reduce the intrinsic inequity of the system, such as those explored in the co-evolutionary behaviour section (6.4); a simple role reversal would solve nothing.

6.2 Rewriting the model to use decision rules

Once the model had been rewritten to use production rules rather than only ordinary Java code, it had to be validated against the original Java model to ensure that it was equivalent. A summary of these tests is given in Appendix A, section 9.3. These tests

established that the models produce similar enough results to allow us to reasonably contrast experiments run in one model with those run in the other. Ideally, all of the preceding experiments should have been re-run in the Drools model and used above instead, to allow more accurate comparison when looking at the results of the extensions, but due to time constraints, this was not possible.

All experiments described from this point onwards were run in the Drools model.

6.3 More than two schools

Note that as described in section 3.2.2, the Java model does not use application rounds, so the random order in which schools consider their applications can affect the results. This is because later schools will give equal precedence to first-round applicants and those who were rejected by earlier schools, and therefore just picked this school as their second choice. This can have a substantial impact on the results, most noticeably in the Java model's version of the four-school class sensitive scenario:

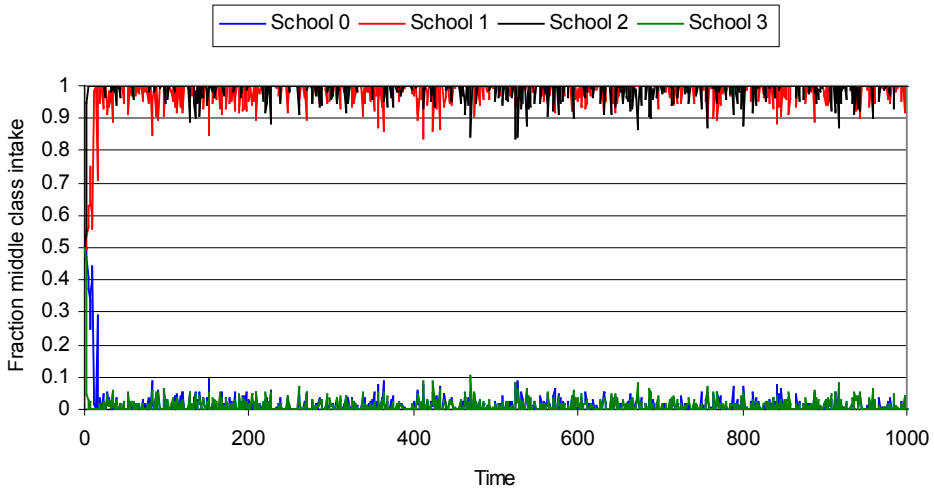


Figure 6.3 – Java model's version of four class-sensitive schools. It is incorrect because it does not use application rounds to order application precedence.

This should be contrasted with Figure 6.5, the correct version. Due to this bug, all multi-school experiments were run in the Drools model.

Room and Britton's model can be extended to cover any number of schools, as explained in section 3.2.2. Unfortunately, this alteration of the parental preference equations invalidates the use of their various predictive equations – Room and Britton restricted their model to only two schools for the very reason that the equations become unmanageable with arbitrary numbers of schools.

We must therefore apply commonsense reasoning to validate the results. We first consider the class sensitive case, with a very high middle-class preference, to make it easier to make predictions. With three schools, it is obvious what is likely to occur: a three-tiered system where all the middle-class families first apply to the leading school; those who are rejected apply to the middle ground school; and those who remain, if any, must go to the bottom-rated school. Based on this reasoning, we could naïvely predict the following in the n school class sensitive case:

Top school: $\min\{\theta n, 1\} = F_1$,

Second-highest school: $\min\{\theta n - F_1, 1\} = F_2$,

Third-highest school: $\min\{\theta n - F_1 - F_2, 1\} = F_3$,

and so on, where F_x refers to the middle-class fraction of the intake of the school in leading position x. However, this is too simplistic: the working-class parental preference, which had no significant effect in the two-school model, now affects the results as well. This is because working-class applicants applying in round 1 are given precedence over middle-class applicants applying in round 2. If some working-class pupils apply to the second-best school in round 1, there may not be sufficient places left for all the middle-class applicants who were rejected from the top school in round 1 due to lack of space. This would only be prevented from occurring in a situation where the top school was preferred with 100% likelihood by both social classes. This would force a kind of repeated two-school scenario between the top school with places left and the rest, all massed together with a 0% likelihood of being chosen by either social class unless the child was rejected by the top school, when the next 'top school' would emerge from the mass.

It also seems possible that the more schools are in play, the greater deviation from these predictions could be, since there will be less differentiation between schools, not just a 100%/0% polarisation, and therefore the parental preferences will have less of an influence. This is predicted to be similar to the two-school cases with weak parental preferences, such as that shown in Figure 5.11.

The multi-school experiments were only run for 200 years, because running 200n agents in the Drools model takes such a long time (see section 3.3.5). Each school was allocated 200 places as before.

6.3.1 Three class sensitive schools

This experiment proceeded exactly according to our predictions:

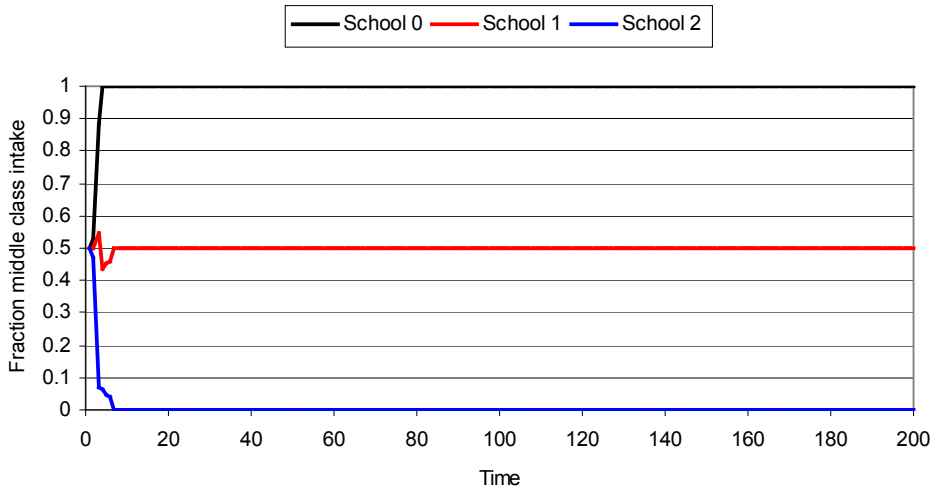


Figure 6.4 – Three schools, class sensitive, with a middle-class and working-class preference of 20.48, and a 0.5 fraction middle-class population.

Figure 6.4 shows the result of using a working-class preference identical to the middle-class preference. This scenario is simple enough to correspond to our naïve equations exactly. If instead the working-class preference is set very low, to 0.01, the overall pattern

remains the same, but the middle school no longer receives an intake of precisely 0.5. Rather, on average, the intake is 0.49, falling as low as 0.4 in one particular year. This is because roughly 1/3 of working-class families now apply to the middle school in the first round (having little preference), which guarantees them a place there, since all middle-class families apply instead to the leading school in the first round. When slightly more than 1/3 of working-class parents pick the middle school, due to chance, less than 100 places will remain for the 100 second round middle-class applicants, and so some must attend the bottom school.

Thus, our prediction that the working-class preference would have an interfering effect holds in the three-school case.

6.3.2 Four class sensitive schools

If the naïve formulae were correct, we would expect to see a pattern of two schools of 100% middle-class, and two schools of 0% middle-class. However, as already established, this is too simplistic; and it already falls down when we move from three to four schools.

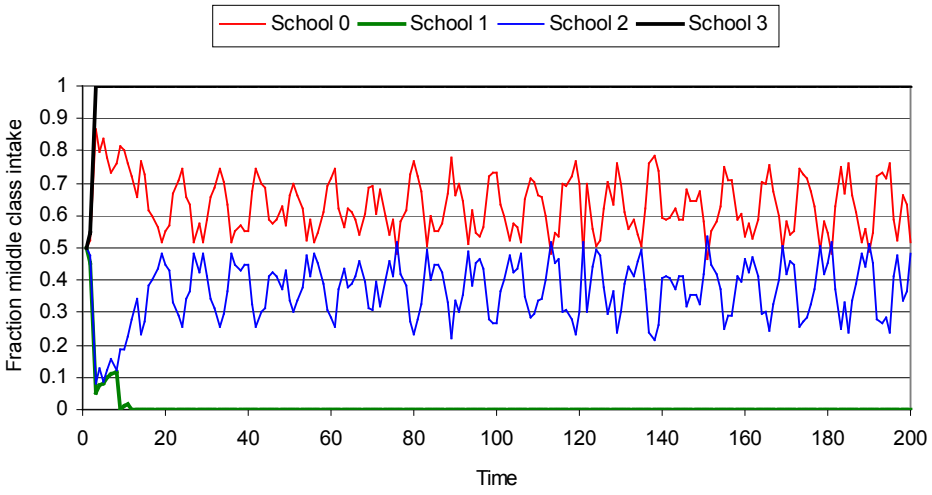


Figure 6.5 – Four schools, class sensitive, with a middle-class and working-class preference of 20.48, and a 0.5 fraction middle-class population.

Examining the choices made by agents at run-time explains how this result is generated:

- Round 1: During a typical year, only around 85% of families will choose school 3, the top school, as their first choice school, since there are other attractive options. However, the remaining 15% will all pick school 0, the second best school; it is as though the other two schools do not exist, like an ordinary two-school scenario. This still easily allows school 3 a 100% middle-class intake, since there are 400 middle-class students in total, and 200 places per school. Around 60 parents of each social class apply instead to school 0, and so it fills circa 120 places.
- Round 2: The only families remaining are now those who were rejected by school 3. This will be approximately 140 middle-class and 340 working-class families. With school 3 removed from consideration, a two-school scenario is again played out, this time between school 0 and school 2, with school 0 favoured by approximately 70% of applicants. The differentiation is less than that in the first round, because the two

schools are separated by less. School 0 receives around 98 middle-class applications; it can therefore fill its remaining 80 places with middle-class applicants, gaining a 70% middle-class intake. School 2 receives around 42 middle-class applications, and around 102 working-class ones, so it fills 144 places.

- Round 3: The 18 middle-class applicants who were rejected by school 0 in the last round remain, as well as 238 working-class applicants. We are again in a two-school scenario between the only remaining schools: school 2, favoured by 100% of applicants; and school 1, chosen by none. School 2 is able to accept all 18 middle-class applicants, resulting in a middle-class intake of around 30%.
- Round 4: School 1 receives all children who were rejected from the other three schools; they are all working-class.

The most interesting thing to note from this walk-through of the agents' decision process is that the problem has been reduced to an iterated two-school scenario; this is not readily apparent from merely looking at Figure 6.5.

Using a working-class preference of 0.01 instead of one equal to the middle-class preference introduces more randomness into the outcome:

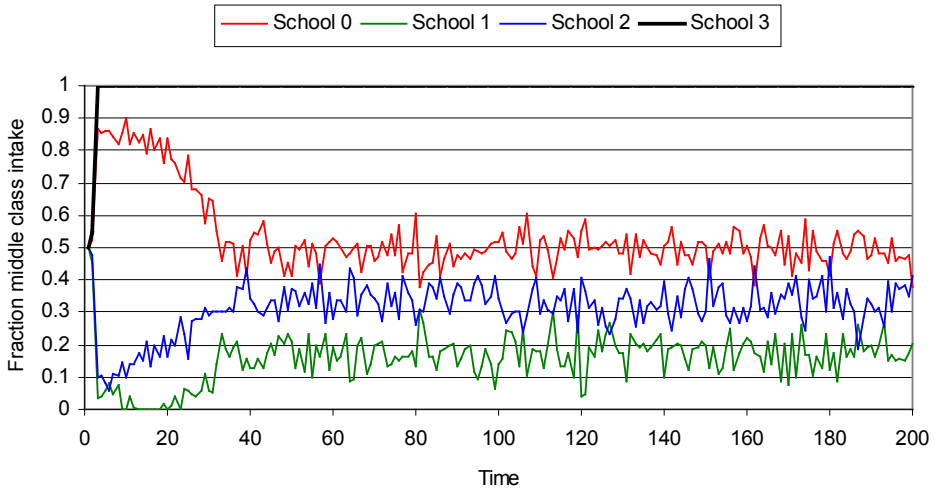


Figure 6.6 – Four class sensitive schools. Using a working-class preference of 0.01 with a middle-class preference of 20.48 results in more 'randomness' as more working-class families take up less desirable, yet not totally undesirable places in the first round, leaving some middle-class families who missed out on the most desirable places only the least desirable options.

6.3.3 Five class sensitive schools

Running a five-school simulation with both parental preferences set to 20.48 as before led to the same pattern as the three-school scenario, with two schools at 1, two schools at 0, and one school in the middle at 0.5. The only difference is that every so often, the middle school would obtain somewhat less than a 0.5 middle-class intake (0.44 in the most extreme case), and one or both of the two bottom schools would receive the 'missing' middle-class applicants.

Delving into the details of agents' decisions shows that they no longer reduce it to an iterated two-school scenario. In the first round, around 99% choose one of the two leading schools, but 1% picks the middle school. In the second round, 82% of those who were rejected from either of the leading schools will apply instead to the other leading school (a guaranteed rejection since it has no places left; the agents are not sophisticated

enough to reason that such a popular school is not a wise second choice). The other 18% apply instead to the middle school, and are all accepted – both middle- and working-class applicants, since there are sufficient places left. In the third round, those middle-class applicants who could not find a place in either of the leading schools all apply to the middle school, leaving it to fill the remaining places with working-class applicants. In round four, the remaining working-class applicants choose between either bottom school arbitrarily.

The anomalous years where the bottom schools receive one or more middle-class pupils occur when an unusual number of working-class students pick the middle school in the first or second round, causing it to fill up earlier than usual. All this leaves the middle-class applicants who could not get into either of the leading schools is one of the bottom schools. The reason we did not see this phenomenon in the equivalent three-school scenario (with equal working- and middle-class preferences) is that the leading school is preferred by every single agent in the first round; there is no room for statistical spikes in a 0% probability.

Setting the working-class preference very low (0.01) causes the distinction between the middle school and the bottom two schools to blur to a much greater degree, even generating an extra level of differentiation where one of the 'bottom' schools is ranked higher than the other. This is due to the middle school's intake being significantly 'diluted' by working-class applicants in the first and second round every year, rather than just occasionally as above. Figure 9.16 and Figure 9.17 in Appendix A provide a visual comparison.

6.3.4 Six or more class sensitive schools

Scenarios with six to ten schools are depicted in Appendix A, Figure 9.18 to Figure 9.22. It is not the case that the ten-school scenario, for example, is the same as two five-school scenarios combined, but the general pattern remains the same, with a distinction found between odd and even numbers of schools. Of course, having, say, five or more schools in a single catchment area is probably generally unrealistic, save in a large metropolis. Nevertheless, it is interesting to consider the potential interactions.

6.3.5 Three class blind schools

We now consider multiple class blind schools. To make the diagrams easier to read, we focus on the most acute differentiation, where the middle-class preference is very high and the working-class preference is very low (20.48 and 0.01 respectively, as before).

The situation here is much simpler, because only the choices of the agents representing the families need be considered. Each school simply accepts as many applicants as it has remaining places in every round, without preference.

This results in the following three tiers:

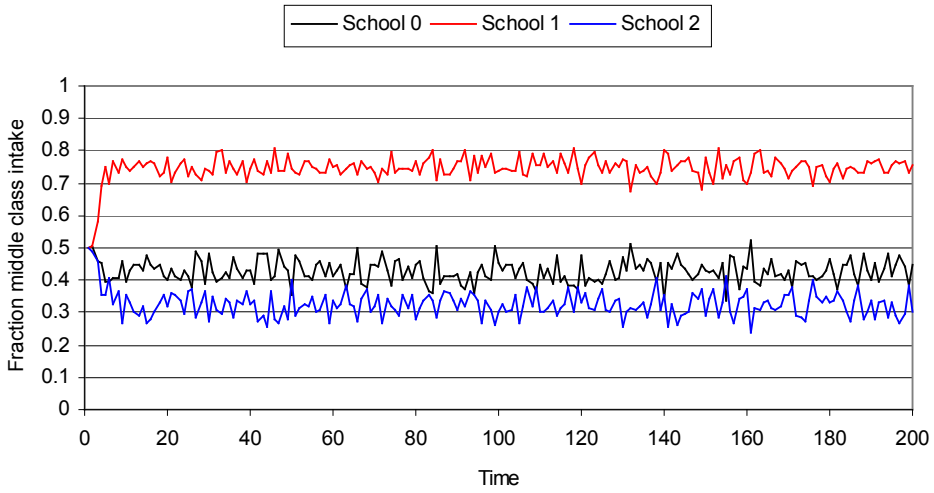


Figure 6.7 – Three class blind schools, with a middle-class preference of 20.48 and a working-class preference of 0.01.

which are essentially a less pronounced version of the three-school class sensitive scenario.

This outcome is readily explained by walking through the application process of a typical year:

- Round 1: 98% of middle-class applicants apply to the top school, school 1, alongside 34% of working-class applicants. School 1 therefore fills all its places in round 1, picking among its applicants at random, resulting in a middle-class intake of approximately 74%. The remaining 2% of middle-class applicants apply instead to the middle school, school 0, alongside 33% of working-class applicants, and are all accepted. School 2 accepts the remaining 33% of working-class pupils.
- Round 2: All of the approximately 144 middle-class pupils who were rejected by school 1 now apply to their second choice, school 0, alongside 25 working-class applicants (50% of those who still seek a place). School 1 has only around 93 places left, so it can only accept around 75% of this round's middle-class applicants. It now has a total middle-class intake of 43%.
- Round 3: All those who were rejected from both other schools now apply to school 2, including 65 middle-class children. This gives school 2 a middle-class intake of 33%.

Equivalently to the class sensitive case, this three-school scenario has been reduced to an iterated two-school scenario for the middle-class families. As always with such a low preference, the working-class families consider all schools at once. This is an advantage for them in the class sensitive case, since they would only be rejected from the leading school(s) anyway, so it makes more sense to aim straight for the second-best choice (although the agents are not as intelligent as this – they only avoid favouring any one school, rather than avoiding a school entirely). However, if they make the same choices as the middle-class applicants here, all inequity vanishes, just as in the two-school experiment 5.1.1.

6.3.6 Four or more class blind schools

Increasing the number of class blind schools does not reveal any new behaviour, and so those results are shown in Appendix A, Figure 9.23 to Figure 9.25. They lack even the

distinction between an odd and an even number of schools shown in the class sensitive case.

6.4 Co-evolutionary behaviour

The co-evolutionary behaviour, as described in section 3.2.6, was examined next. These were the most interesting experiments: each one built cumulatively on the results of the next, since the rules of the previous experiment were added to, not replaced. This led to interesting interactions.

6.4.1 Preference for class blind schools

Rules: none.

The newly introduced preference of families for class blind schools, now competing with their preference for high-performing schools, was first tested without the addition of any 'co-evolutionary' rules. The experiment looked at varying values of class blind preference mean: the logarithmically increasing sequence 0, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32 and 0.64, as well as the maximum possible value of 1. Each preference mean was tested with standard deviations of 0 and 0.2. One class blind and one class sensitive school were used, to force children to choose between a class blind and a well-performing school (a class sensitive school will always dominate a class blind school in the absence of other factors, and assuming sufficiently high parental preferences). The weighting used for a school's league table rating was set high to 0.8 (with a standard deviation of 0.2), since it was felt that this would still be the most significant parameter. This left a weighting of 0.2 on average for class blind preferences.

The predicted effect of this competing school attribute was to curtail the leading school's trajectory to the top, preventing it from reaching a 100% middle-class intake. A larger class blind preference should result in a smaller mean inequity between the schools. Without the class blind preference, an inequity of 1.0 would usually result; it should equivalently be found now with a class blind preference of 0 and a standard deviation of 0. The effect was as predicted, as seen below, where the result of not using the class blind factor is overlaid onto the outcome of using a class blind preference of 0.32:

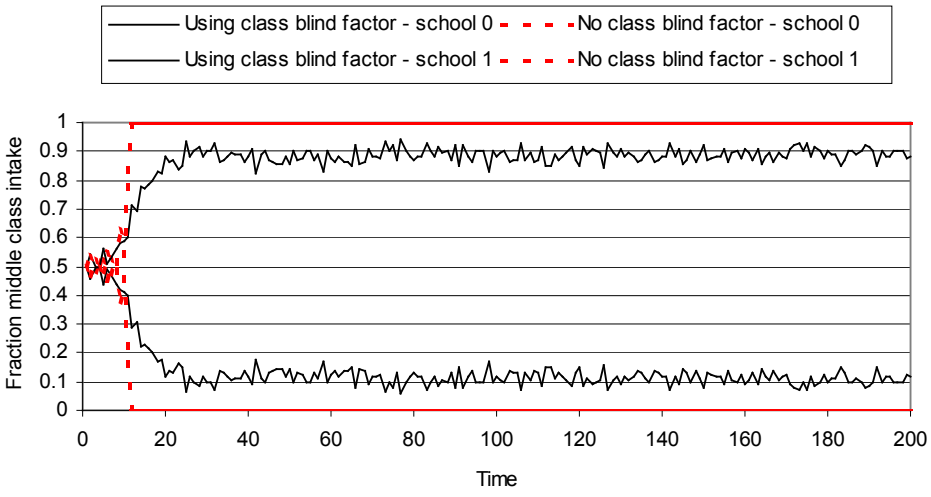


Figure 6.8 – Two runs overlaid: In dotted red, the standard result of using one class blind and one class sensitive school with a high middle-class preference of 20.48, and a low working-class

preference of 0.01. In black, the same scenario using class blind preferences generated from a normal distribution with a mean of 0.32 and a standard deviation of 0.2.

Whether a standard deviation of 0 or 0.2 was used did not make a significant difference to the stable state settled upon (measured by mean inequity between the schools):

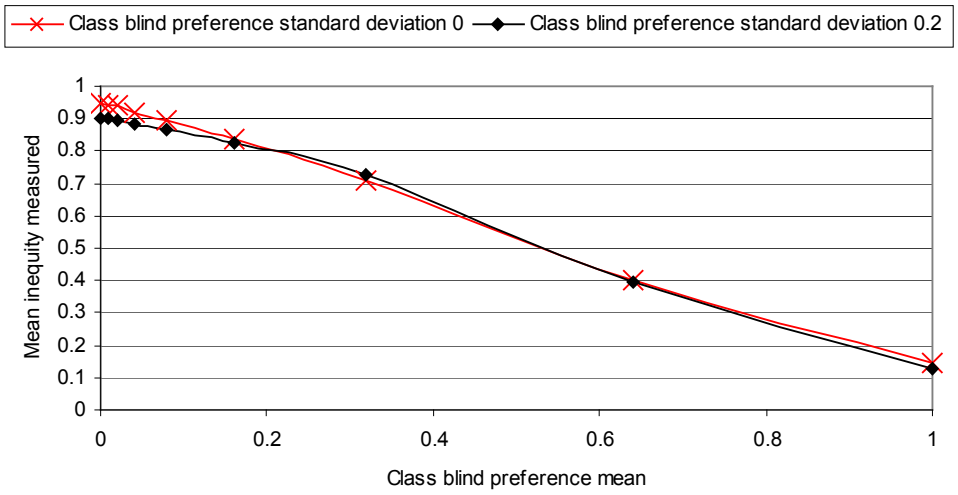


Figure 6.9 Scatterplot showing how an increasing class blind preference mean parameter causes the average inequity to fall when using one class blind and one class sensitive school.

What standard deviation was used for calculating each child's class blind preference only had a significant impact when the class blind preference mean was close to 0. This is logical, since when the class blind preference is close to 0, very little impact would be seen; using a standard deviation of 0.2 would make many more parents care about class blind schools than sticking to a preference of 0, 0.01 or 0.02 exactly. Since all preferences < 0 are truncated to 0, this effect is not averaged out by values falling below the mean, as in the cases with a large class blind mean.

The experiment was also run with four schools, two of which were class blind, and two class sensitive. For low class blind preferences (up to 0.32), the top class blind school assumed a leading position with a middle-class intake of around 75%, with one class sensitive school rated marginally above the two remaining schools. For higher class blind preferences, the two class blind schools both led with a middle-class intake of around 65%, while the two class sensitive schools assumed differentiated positions of around 45% and 25%. The class sensitive schools' strategies were disapproved of too much to be preferred by children, and yet their league table positions still had an effect when given the choice of only those two.

6.4.2 Children aiming to reduce polarisation

Rules: section 3.2.6 rules 3.a and 3.b are introduced here.

The first rules to be tested were the ones used by children to attempt to curb a tendency towards total polarisation between schools. This was motivated by the reasoning that interest in a class blind school would be rather low when all schools were approximately equal in social class intake anyway, and that conversely it would be exaggerated if it was felt that social inequity was a severe problem in the community.

The expected effect of the first rule, which decreases the children's class blind preference by 20%, was to diminish the initial difficulty in beginning the trajectory towards polarisation compared to the previous experiment. This was exactly the case, as shown

below where we again see the class blind preference mean of 0.32 as in Figure 6.8, overlaid onto the result of using the child rules:

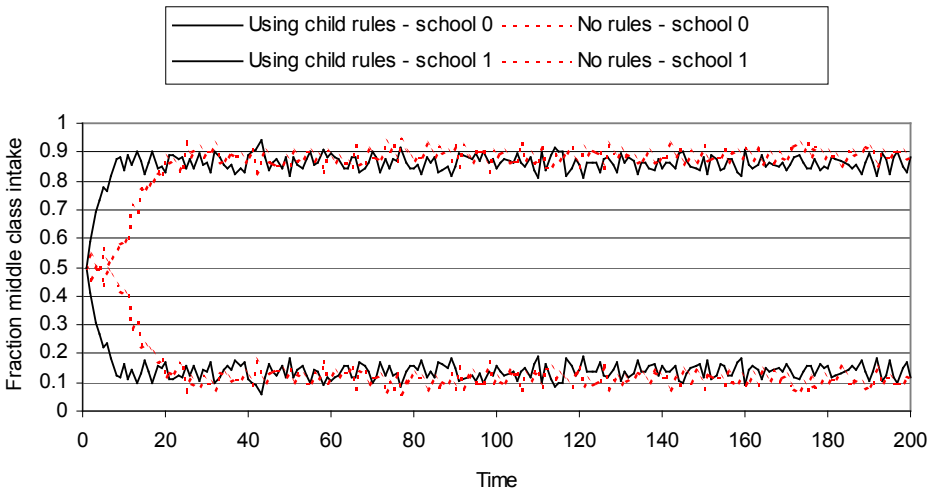


Figure 6.10 - Two runs overlaid: In dotted red, the result of using one class blind and one class sensitive school with a high middle-class preference of 20.48 with a low working-class preference of 0.01, and a class blind preference mean of 0.32 (0.2 standard deviation). In black, the same scenario using the child rules to adjust the class blind preference depending on the environment.

The experiment used the same combinations of class blind preference mean and standard deviation as the previous one. One class blind and one class sensitive school were also used as in the previous experiment.

As before, the inequity was diminished with increasing preference for class blind schools, and whether a standard deviation of 0 or 0.2 was used did not make a significant difference.

Just as in the previous case, the second rule was also expected to freeze the polarisation trajectory once the threshold for the increased preference for class blind schools was passed. This was defined in the rule as a school having 170% of the population average as a middle-class intake. How low the inequity is capped should depend on the strength of families' preferences for class blind schools. However, the inequity was expected to be somewhat reduced compared to the previous experiment, because the children perceive the near-polarisation and work against it[31] by altering their preferences, increasing them by 20%. The difference was found to be less than intuitively expected (although of course no concrete value could be predicted); in Figure 6.10, the difference in mean inequity is only 0.01. On average, the mean difference in inequity between the two sets of runs was 0.02. It therefore seems that the child rules did not succeed very well in their aim of reducing the inequity between schools; in fact, they accelerated the trajectory towards that stable state! Nevertheless, the rules were left in for the remainder of the experiments, in case any interesting interactions were found, and because it could still make sense for an individual household to reason in this way.

6.4.3 Unsuccessful school adopts class blind niche

Rules: section 3.2.6 rules 3.a and 3.b as before; 4.a is introduced here.

[31] Their individual motivation is not really to counteract the polarisation effect – rather, they do not wish to attend a completely polarised school, so they become more interested in other, more suitable schools – but as a strategy held by all children, it could still be said to be a collective 'unconscious' motivation.

Next, the rule that allows a class sensitive school performing poorly in the league tables to enter a class blind niche position to win middle-class applicants was added. This was tested with two class sensitive schools, again using the extreme 20.48 middle-class and 0.01 working-class preferences to magnify the effects. Due to the Matthew effect (Room and Britton, 2006b), the school that gained an initial small advantage quickly moved up the league table rankings. Once the other school's league table rating had fallen past the threshold of 90% of the population average, it would give up trying to compete with the other school in the league table, and instead change its strategy to class blind. The results were very similar to those of the previous experiment; a comparison is shown below:

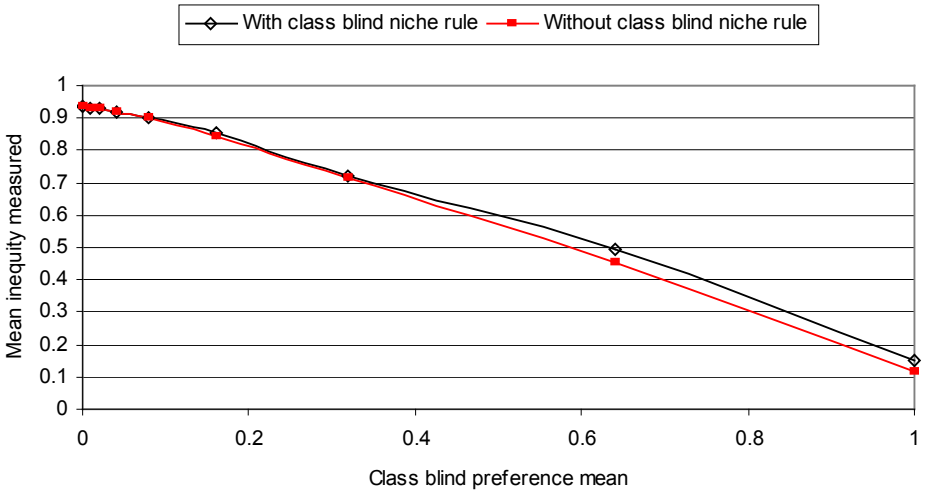


Figure 6.11 – Comparison of the mean inequity measured between a class blind and a class sensitive school, and between a run starting with two class sensitive schools with the 'failing' school switching to a class blind strategy during year 3.

Only the results for a standard deviation of 0.2 are shown, since a standard deviation of 0 showed the same pattern. The mean inequity measured between the two schools was higher using the class blind niche rule than without it for high class blind preferences. Comparing the two runs reveals a very simple explanation for this:

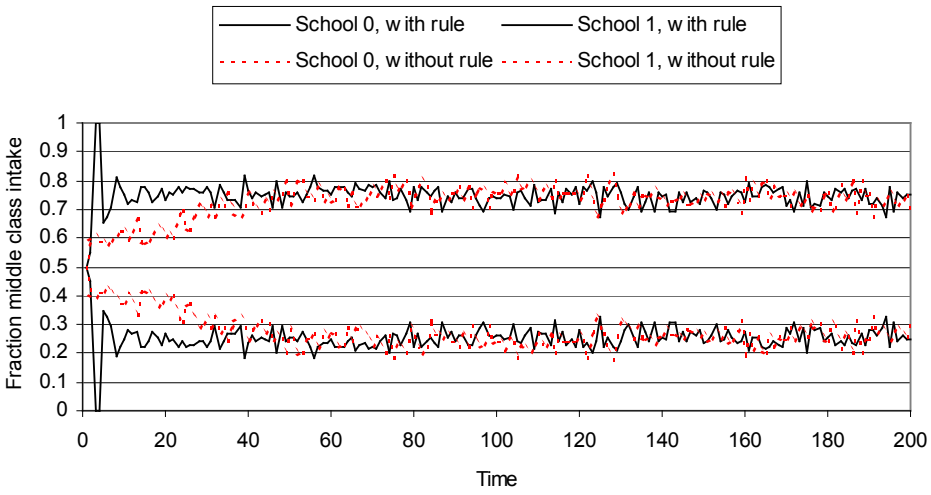


Figure 6.12 – Two runs using a class blind preference of 0.64 are superimposed here; the solid black one uses the niche rule, and the dotted red does not (it is a result from experiment 6.4.2).

Because the middle-class preference for high-performing schools is so high (20.48), the downwards spiral experienced by the failing school happens so rapidly that the switch to a class blind strategy in year 3 cannot prevent a year in which total polarisation occurs. This initial spike before stabilisation causes the increase in measured inequity when using the class blind niche rule. It is not as noticeable with high values of inequity.

6.4.4 Popular class blind school becomes class sensitive

Rules: section 3.2.6 rules 3.a, 3.b and 4.a as before; 4.b is introduced here.

We now examine the rule that reasoned that a sufficiently high-performing class blind school could become class sensitive, thereby removing the cap on its league table rating as calculated in section 5.1 for class blind schools. This is effectively the other side of the coin from the previous rule we investigated. However, while no harm can be seen in the previous rule – if the school did nothing, the worst case scenario of 0% middle-class intake would be reached in any case, so adopting a class blind niche could only improve matters or at worst leave them the same – this rule could potentially backfire if the population preferred class blind schools so much that the school became unpopular.

As before, an extreme middle-class preference of 20.48 and a working-class preference of 0.01 were used, to 'exaggerate' the interactions, allowing easier identification of patterns. This time, two class blind schools were used to begin with, since this is the scenario where the rule would be applied. The same set of class blind preferences were used to initialise the normal distribution as previously.

'Backfiring' of the strategy was only seen in the most extreme case, with a maximal class blind preference mean (1.0). Here, the same 'game' was played over and over again, with the schools switching roles each time: the popular school switches to a class sensitive strategy, but this makes it so unpopular that it loses favour and the other school becomes more popular instead. Eventually, it becomes so popular that it becomes class sensitive as well, whereupon the original leader immediate adopts a class blind niche, which allows it to ascend into popularity again, and so on. Below we see a graph of this, overlaid onto the result of not using the class sensitive rule:

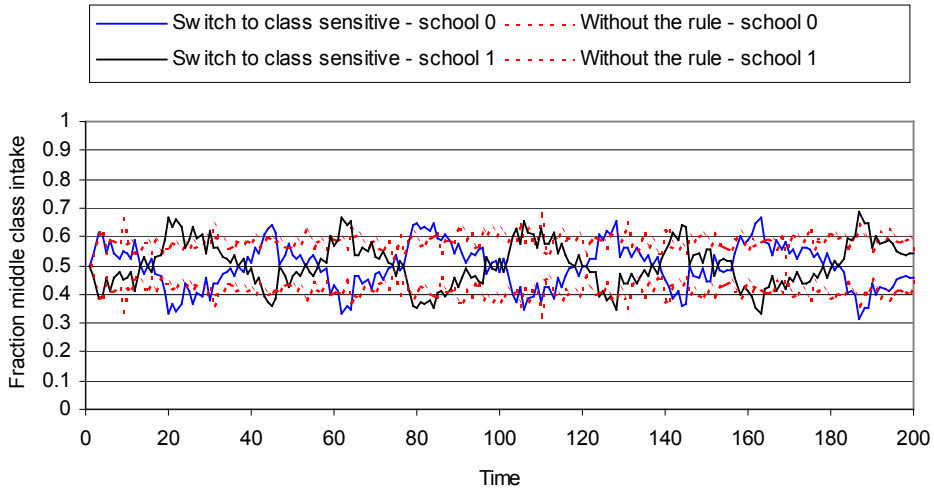


Figure 6.13 – Two runs overlaid: In dotted red, the run using a middle-class preference of 20.48 with a working-class preference of 0.01 and a class blind preference mean of 1.0 (standard deviation 0.2). In black and blue, the same parameters were used with the class sensitive rule. Note that the two runs follow the same trajectories until the end of year 5, when the leading school becomes class sensitive due to its popularity.

However, in every other case, the strategy switch was advantageous for the leading school, indicating that it is a sound rule to follow (since a class blind preference of the maximum value, 1.0, is probably not realistic). It resulted in a greater inequity between schools, as shown below:

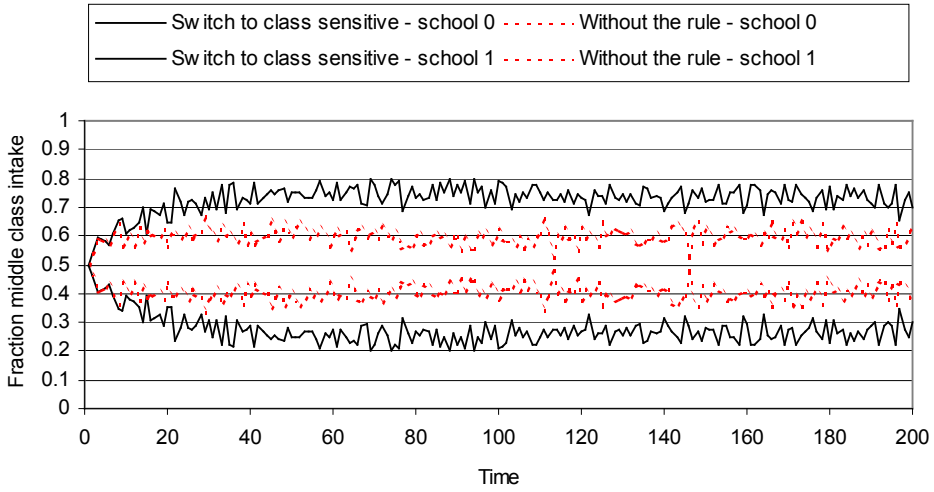


Figure 6.14 – Two runs overlaid: In dotted red, the run using middle-class preference 20.48, working-class preference 0.01 and a class blind preference mean of 0.64 (standard deviation 0.2). In black, the same parameters were used with the class sensitive rule. As in Figure 6.13, the leading school switched to a class sensitive strategy at the end of year 5.

The experiments were also run with three and four schools, again assigning all schools the class blind strategy at the start.

For three schools, a clear leader immediately emerged, became class sensitive and maintained a 100% middle-class intake except for blips for a mean class blind preference of 0.64, and significant troughs for a mean preference of 1.0. The other two schools remained class blind and formed a local two-school class blind scenario for the remaining applicants. The inequity between those two lessened and then disappeared with increasing class blind preference means, because the league table position of a school became less important. An example is shown in Appendix A, Figure 9.27.

In the four-school scenario, each parameter combination produced a similar result: the popular class sensitive school achieved 100% middle-class intake throughout, leaving the remaining three class-blind schools to share an equal intake of around ⅓ middle-class. However, the cases where the class blind preference was very low allowed a slight leader to emerge among those remaining three schools. This differentiation no longer occurred for class blind means of 0.32 and higher, for the same reason as in the three-school case.

6.4.5 School becomes class sensitive to break out of stable, undifferentiated state

Rules: section 3.2.6 rules 3.a, 3.b, 4.a and 4.b as before; 4.c is introduced here.

The final rule concerned with schools changing their strategy was one designed to upset the stable class blind state where the parental preference is insufficiently high to produce differentiation between schools. A middle-class preference of 1.28 with a working-class preference of 0.01 was used; this is not sufficient to generate differentiation between schools by itself. The simulation was initialised with two class blind schools, and then a random race condition determined which school took the risk of becoming class sensitive.

The rule only fires when no school's middle-class intake is further than 10% away from the population average.

As usual, the simulation was run with a logarithmically increasing class blind mean, from 0 to 1. The rule fired at the end of year 1 in every case, as expected, but it only had an effect in the first case, with a class blind preference mean of 0. Even then, it took a while to take effect:

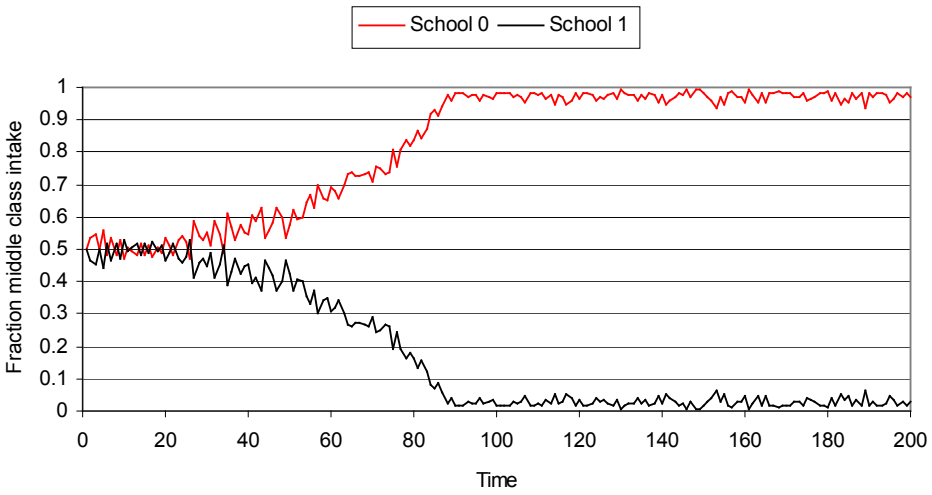


Figure 6.15 – School 0 applies the 'risky' rule in year 1, allowing it to ascend to near-polarisation, but only after a relatively long time. The class blind preference mean is 0.

This long delay before the change could take effect – since the class sensitive school first needed to be 'lucky' to build up a sufficiently large advantage to differentiate itself permanently – could explain why the other cases did not exhibit this outcome; the stronger class blind preference prevented the class sensitive school from gaining in popularity. In fact, with a mean preference of 0.64 and 1, the class sensitive school sees such a decline in popularity because of its strategy switch that it is forced to revert to a class blind strategy to save itself. However, the other school is no more aware of the children's strong preferences than it is, and so it then falls into the same trap: its gain in popularity causes it to boldly become class sensitive, after which it is disfavoured in turn, and so on. A more intelligent set of rules would allow schools to recognise this extreme scenario and avoid using this strategy as it is obviously counter-productive in this case. With the somewhat lower class blind preferences 0.16 and 0.32, the class sensitive school merely takes on a slightly less popular role for the duration of the run.

The range 0.01 – 0.08 therefore seemed like a grey area – would polarisation occur after all, if we waited long enough? The 0.01 case was re-run ten times with different random seeds, and the polarisation effect emerged at varying intervals in four cases, which supports this theory. The 0.04, 0.08 and 0.32 cases were also re-run ten times each in order to verify that no differentiation was emerge; this was confirmed.

6.4.6 School specialisms

Rules: section 3.2.6 rules 3.a, 3.b, 4.a, 4.b and 4.c as before; no new rules are introduced in this experiment.

Adding a third attribute to schools, their specialist subject, adds much potential for interference between rules, so the effect of using specialisms was first investigated without the use of any additional rules. School specialisms are detailed in section 3.2.6,

but the key points are that each school may hold at most one specialism, and it cannot change specialism more than once every ten years.

The case of schools having the same specialism was not examined in detail, since it does not do anything other than slightly weaken the effect of other differences, just as the addition of a class blind preference lessened the impact of school league table ratings. Therefore, each school was given a distinct specialism, and the only variable parameter was the number of class blind schools (0, 1 or 2). A class blind preference mean of 0.32 was used, having been previously established as a non-extreme value, with the usual standard deviation of 0.2. As before, a league table weighting mean of 0.8 was used, also with a standard deviation of 0.2. The remaining weighting was divided equally between a school's specialism and its strategy (class blind or not), which is somewhat arbitrary, but any division would merely be a guess in the absence of field data, The middle- and working-class preferences were left at 20.48 and 0.01 respectively.

We would expect the effect of a high class blind preference to be lessened from now on, since it is now given a lower weighting. Apart from that, no difference is expected in this experiment from those found with the same rules, i.e. those of experiment 6.4.5. This is because children are not biased towards any particular specialism in the simulation (on average), so no specialism is really any better in the long term, even if it is significantly preferred by a particular year group due to a statistical anomaly.

The results were as expected. In the 0-, 1- and 2- class blind school cases, the mean inequity was 0.0349, 0.0299 and 0.0295 less respectively than when school specialisms were not used. Additionally, when two class blind schools were used, the initial plunge towards polarisation that is later corrected became a gentler ascension when specialisms were introduced (see Appendix A, Figure 9.26). This explains the greater difference between the mean inequities in this scenario. It was not predicted by the experimenter, who failed to consider every condition, but this again illustrates the value of simulation, especially when so many behaviours are interacting and interfering with each other that prediction becomes demanding.

6.4.7 School specialism rule to avoid duplicate specialisms

Rules: section 3.2.6 rules 3.a, 3.b, 4.a, 4.b and 4.c as before; 4.d is introduced here.

Next, a rule was added to encode the behaviour that a school should switch to another specialism if possible if another, higher-performing school shares the same one. Otherwise, it would not gain any advantage from its specialism: all the children who were interested in it would still favour the higher-performing school (unless perhaps it had a less preferred admissions strategy).

The schools were both initialised with the same specialism to test the effects of the rule. As in the previous experiment, a class blind preference mean of 0.32 was used, with a 0.2 standard deviation; the middle-class preference was 20.48 and the working-class preference was 0.01. Runs were performed with both two and three specialisms to choose from, and with 0, 1 and 2 class blind schools on initialisation. Three specialisms were not expected to produce a different effect from only two, since the schools do not give much thought to which specialism to switch to, only that it is not already taken.

The simulation does not provide a way of specifying exactly which schools should have which specialisms (see section 8.3.6), but fortunately, the usual random seed produced the desired set-up in the two-specialism case. However, for three specialisms, a different random seed had to be used (3). Fortunately, whether two or three specialisms were used had no significant effect (as predicted), and so those results were not required to be compared against any other experiment's.

The rule fired at the same point in each case (at the first available opportunity, i.e. in year 11 since 10 years are required as a minimum duration for a specialism's adoption).

Surprisingly, no significant difference was found for the two-specialism runs when compared to simulations run with the same parameters but without this new rule. This indicates that the impact of giving each child linearly decreasing interest levels in the specialisms, so that year on year 50% of children preferred specialism 1 approximately to degree '1', and the other 50% preferred it to degree 0.5, coupled with the small ~10% weighting given to specialisms, is to produce an even smaller end effect than previously assumed. However, the alternative explanation of there being an error in the program cannot be ruled out either, even though no individual behaviour observed during the simulation could be identified as anomalous.

6.4.8 School specialism rule to sabotage competitors with duplicate specialisms

Rules: section 3.2.6 rules 3.a, 3.b, 4.a, 4.b, 4.c and 4.d as before; 4.e is introduced here.

Finally, the rule was added in which high-performing schools try to inflate their advantage even further by sabotaging their less well-performing competitors' niche positions. They do not want to take away a class blind advantage because that would cap their maximum league table rating, but there is no real disadvantage to changing different specialism. Since the other school that shares the specialism has an inferior league table rating, the vast majority of children who previously would have chosen the specialist niche school due to their academic interests will now turn to the high-performing school instead.

Again, two and three specialisms were examined, and 0-, 1- and 2-class blind school scenarios, with middle- and working-class preferences of 20.48 and 0.01 respectively.

As in the previous experiment, the number of specialisms used had no significant effect: every 11 years the 'failing' school would switch specialism to try to adopt a niche as before, but the leading school now immediately followed suit, preventing it from gaining any advantage. Whether one or the other 'unused' specialism was used for this in the three-specialism case obviously made no difference.

Compared to runs where the rule was not used, the 'act of sabotage' had only a very mild effect: an increase in inequity of 0.0013 on average. This was less than expected, but on further investigation, it seemed reasonable for school specialisms not to have a very strong effect: they only receive a weighting of 10% on average, and each child does not only foster a single interest: it has a linearly decreasing range of interests. Nevertheless, it is difficult to put a value on the expected difference in inequity, and accordingly difficult to say whether this experiment conforms to expectation or not. The strongest statement that can be confidently made is that the reasoning made by the schools and the children can be followed.

6.4.9 Children excluding irrelevant factors

Rules: section 3.2.6 rules 3.a, 3.b, 4.a, 4.b, 4.c, 4.d and 4.e as before; 3.c, 3.d, and 3.e are introduced here.

The last rules added were for children to exclude factors from consideration that were irrelevant, such as the class blind preference if all schools shared the same strategy, or their academic interests if all schools had the same specialist subject.

This is arguably an implementation issue, but the effect on the results is so profound that it was included as an experiment. The new rules increased the 'rush' towards a stable, inequitable state even more so than the children's rules for adjusting their league table preference (introduced in 6.4.2) alone. The experiment was run with 0, 1 and 2 class blind

schools to begin with; the effect was most pronounced in the 0-class blind school case, when the popular school switches to a class sensitive strategy:

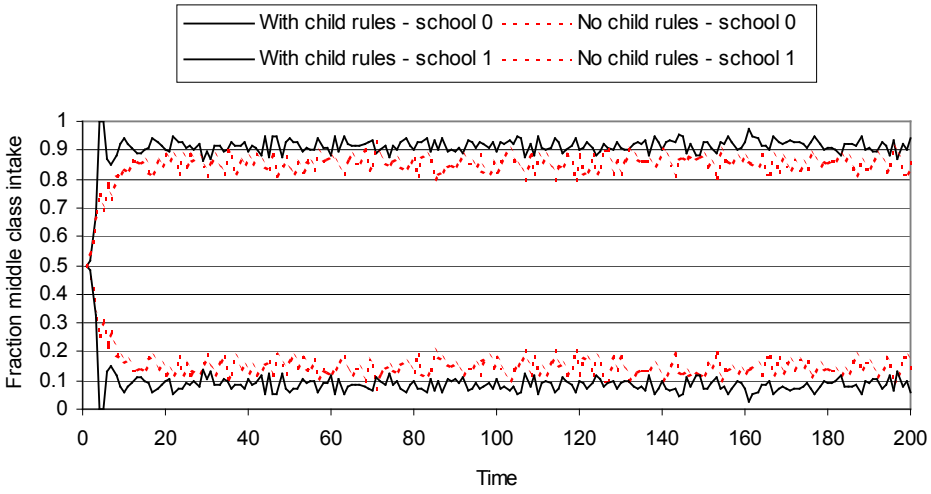


Figure 6.16 – Two runs superimposed: In dotted red, two (initially) class blind schools, with two specialisms to choose between, and middle- and working-class preferences of 20.48 and 0.01 respectively, alongside the usual 0.32 mean class blind preference, using all rules bar the child factor elimination ones. In black, the same experiment with the child rules to ignore irrelevant school choice factors.

The significant increase in mean inequity seen above was also seen in the other simulation runs.

In the three-school case, run with the usual increasing class blind preferences, the accelerated move towards the stable state was also very pronounced (compare Figure 9.27 to Figure 9.29 in Appendix A). The acceleration was less obvious in the four-school scenario, because it already happened so quickly before (compare Figure 9.28 to Figure 9.30 in Appendix A).

6.4.10 Reducing initial inequity

Having analysed our set of 'co-evolutionary' rules, we would like to see whether they help (or possibly even hinder) a reduction of initial inequity in a simulation.

Using all the rules introduced here, an experiment was run using an initial inequity of 0.8, and one of 0.1 (the usual initial inequity is of course 0 – the schools begin as equals). This was tested with two class blind schools and a middle-class preference of 5.12, paired with working-class preferences of 0.01, 0.04, 0.08, 0.16 and 0.32 to produce varying degrees of intrinsic inequity. The league table rating was weighted at 0.8 with a standard deviation of 0.2 as usual; both the class blind preference mean and its standard deviation were set to 0.2; and two possible specialisms were used.

The same experiment was also run without using any rules or co-evolutionary behaviour at all. The results are shown below for the example case of a working-class preference of 0.32; they were equivalent for the other working-class preferences:

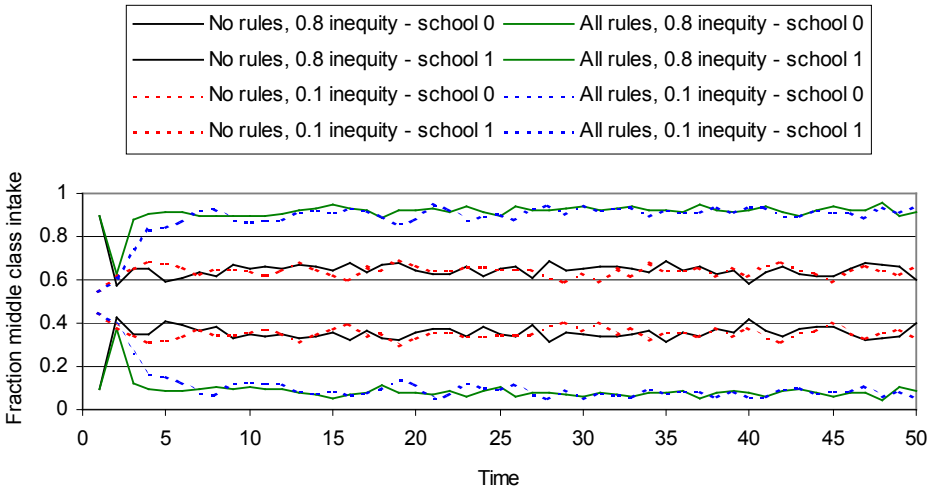


Figure 6.17 – This figure shows four runs in total. The dotted lines are runs beginning with initial inequity 0.1, while the solid ones represent runs with an initial inequity of 0.8. The uppermost two and the corresponding lower two schools are the two runs that used co-evolutionary behaviour. The four schools (i.e. two runs) in the middle are those that did not use any co-evolutionary rules.

What Figure 6.17 clearly shows is that the initial inequity used is irrelevant; the system tends inexorably towards the stable state of its intrinsic inequity, described by the school strategies and parental preferences used. This is in line with Room and Britton's predictions for the basic model; it is interesting that they hold for the more advanced model as well.

Of course, this does not imply that the agents cannot alter the intrinsic inequity of the system: the main reason the co-evolutionary rules produce much greater differentiation between schools is that one school becomes class sensitive, enabling it to climb higher in the league tables. Instead, it indicates that the initial environmental conditions of a simulation are inconsequential.

7 Discussion of Results

7.1 Validity of results

In chapter 5, we showed that the simulation was essentially able to replicate Room and Britton's results (2006b), if allowances are made for stochastic effects which would also be present in the real world. The simulation is perhaps a truer replica of the real life domain than the mathematical model is, since the equations only capture the macro-level effects, missing the fine detail such as local spikes in school league table ratings. We have noted that this type of local statistical anomaly, while probably occurring commonly in the small sample sizes found in the real world in a given catchment area's population, can have a long-term impact on the macro effect, slowing the trajectory towards the inequitable state. However, the eventual stable state remains the same.

Yet it is important to note that there is no proof that the micro-behaviour of the model is at all representative of the real world. We can say that it is probably similar to the type of behaviour that would be seen, but the exact details are undoubtedly incorrect. Of course, this was never a major aim of the project; rather, it focused on simulating the underlying mechanisms and trajectories, thereby improving understanding of the problem domain and of this type of system in general. The correspondence (or lack of it) to the real world system was outside the scope of this project. Without the necessary empirical data, it would be counter-productive to try to introduce additional parameters with the sole aim of adding 'realism': the experimenter would be forced to guess at appropriate values, thereby creating further room for error in the results. For example, how would one quantify the preference parents might have for sending all their children to the same school? A full parameter sweep of these new factors would be necessary, which would soon lead to a combinatorial explosion in the parameter space to be explored – and to no real gain.

Of course, adding unnecessary complexity to the model would also make it much more difficult to understand, and require even more computing power. Even at its current modest level of complexity, it is no longer possible to look only at a graph of the league table ratings to infer what behaviours occurred when. Moreover, the model would risk becoming overspecialised: the intention was to look at general co-evolutionary mechanisms that are common to many domains.

It could nevertheless be a valuable exercise to validate the model against empirical data. This should expose any fundamental flaws in the model's behaviours, and would provide insight into how well the agents mimicked their real life counterparts. However, 'docking' a model containing stochastic elements to empirical data is complicated, as discussed in section 2.6.

Although the model's basic features could be validated against Room and Britton's model, the extensions investigated in chapter 6 could not really be validated at all, apart from against the predictions of the experimenter. The experimenter was biased both as the author of the code, and from having seen informal experiments while testing that would have created preconceptions.

On a more fundamental level, the project also suffered a lack of repeated experiments: ideally, every single experimental case would have been run 100 times with different random seeds. Time and computing power constraints did not allow this, but it would be essential for a more ambitious project.

It would also have been preferable to check every experiment for replicability, just in case the model was perfectly replicable in all but a small number of aberrant cases; instead only approximately 10% were checked. This however brings with it again the problem of deciding when enough simulations have been run: a program might produce the same result 9 times out of 10 but deviate on the 10^{th} run, for example. For greater confidence, several different types of computer should be used for these tests, to ensure that the effects of uncontrolled factors, such as the order the JVM iterates through an unordered list, are not being masked by the particular system being used. This was already checked to some degree in this project by using seven different Windows machines, and a powerful Linux server. However, it is impossible to establish replicability with full certainty since we cannot run an infinite number of experiments.

The parameter space could also have been explored more fully; this is especially true of the later, co-evolutionary experiments, when time was running out. It would have been interesting to run more multiple-school experiments with variants of the co-evolutionary behaviour. Stochastic shocks were not investigated for multiple schools at all either. These experiments would have been run, had time permitted.

A finer point is that certain parameters were hard-coded into the model. Specifically, these were a) all 'threshold' values in rules, e.g. specifying that the rule should fire if the school's league table rating rose 10% or more above the average; b) what value children should increase or decrease their class blind preference by, when the appropriate rules fired; and c) the half-life of a stochastic shock, allowing its decay rate to be specified. There was insufficient time to explore the parameter space of these variables, but the fact that this was not done further weakens the validity of the model's results. We cannot claim that the model's behaviour would not have differed significantly if any of these parameters had been different, since it was not tested.

A policy-maker, upon seeing the experimental results, might find the hundreds, sometimes thousands of years simulated to be ridiculous. Firstly, of course schools as we currently know them can hardly be expected to exist in a thousand years' time, or even in a hundred year's time, due to local and national politics; and secondly, prediction of more than a few years or decades into the future is doomed to failure, as any error in the simulation is multiplied on every successive step.

However, the experimental results are not really intended for prediction – rather, they were meant to aid understanding of the fundamental processes at work in the domain. Upon seeing a school's rising trend in the league tables, it is of interest to know whether this will be capped at a certain point, or if it would in theory continue indefinitely, until it had reached a 100% middle-class intake. This helps us to understand the underlying mechanisms. Of course, with extreme parental preferences, such total polarisation can indeed be reached after only a few years, and no doubt, such schools do exist in the UK; but more commonly, the polarisation found in real life, and in the simulation after a 'realistic' number of years, is only partial.

7.2 Key findings

The overarching finding of all the experiments run is that the school choice system always tends towards its intrinsic inequity. This is defined by the schools' strategies and the children's preferences – although these can be changed dynamically when co-evolutionary behaviour is enabled. However, any initial inequity at the beginning of the simulation run is irrelevant; it cannot be sustained if the preferences and strategies do not support it naturally. Even the stochastic shocks could not alter the inequity present in a system; they could only switch the school's roles.

The addition of rules to allow schools to change strategy made a huge difference: a school doing poorly on the league tables could 'save' itself by adopting a class blind niche, maintaining a certain level of middle-class intake that would otherwise have gradually disappeared. Conversely, a popular class blind school could now 'abuse' its position and change to a class sensitive admissions policy, allowing it to move from a relatively high to a 100% middle-class intake, completely excluding working-class applicants. On the whole, the co-evolutionary rules used tended to aid the leading school, increasing the inequity in the system: while they did provide a tool for a less successful school to form a niche, often this niche was only necessary because another school had 'defected' and become class sensitive, taking a huge lead. The general tendency of all the co-evolutionary rules interacting together, in the two-school case, was for there to be one very successful class sensitive school, and one fairly (but not fully) unsuccessful class blind school. Admittedly, this is not as inequitable as the basic case of two class sensitive schools. However, it could be argued that different rules would have produced an entirely different effect, although the adoption of niche positions was at least observed in the field by Lauder and Hughes (1999).

In contrast, specialisms were a disappointing extension. Being given a weighting of, on average, only 10%, and dividing the child's interest levels between several specialisms, meant that the effect of a school's specialism was much diluted. It seemed too weak to have any real effect of significance on the schools' trajectories. It would perhaps have been preferable to give each child only a single subject of interest, just as each school can only have a single specialism. The specialism rules were not especially intelligent in any case: it would have been more advantageous for a school to consider all possible matches and take the most potentially profitable one, rather than being satisfied with the first one meeting the basic criteria. For example, an ethically questionable school could perhaps have sabotaged two of its competitors at once rather than just one. However, there was insufficient time to investigate many multi-school scenarios involving specialisms, which is where this would have come into play, since this was the last extension added to the simulation. Nor was there much time to investigate the effects of using varying numbers of specialisms; but initial results showed that their number was of little consequence, perhaps again due to the weak influence of specialisms in general.

Another criticism that could be made of the school specialism rules is that it is unrealistic for schools to be happy to switch specialism every ten years. The rules should include some concept of the huge overhead involved in such a change: specialist equipment might become obsolete, and major staffing changes could be necessary. It is true that the Specialist Schools Programme devised by the DfES (2003a) requires a school's specialism to be renewed every four years, but the school is expected to retain the old specialism. A second unrealistic aspect of the model is that in real life, the DfES might well reject a specialism proposal that was seen as an act of sabotage towards other schools, or that failed to add value to the community, being already well provided for by another school.

It might also have been more interesting if negative levels of interest in a specialism were possible in a minority of cases. For example, a poor linguist or a dyslexic child might not wish to attend a specialist language school for fear their grades would suffer, having greater emphasis on languages.

8 Conclusion

8.1 Evaluation of the project

The overall aim of this project was to produce an agent-based model equivalent to Room and Britton's equations (2006b), and then to build upon it to deepen understanding of the underlying processes, and to demonstrate the advantages of agent-based simulation. In this, the project was a success.

The simulation was able to replicate all essential aspects of Room and Britton's model; and moreover in doing so, it is claimed that it exposed an oversight in the original model. This was only detected after the initial prototype had been written, which gave the author a much deeper understanding of the trajectories and mechanisms in play than the original equations or textual descriptions had been able to. Thus, it was possible to identify the mechanism that Room and Britton's model did not appear to account for. It would have been much more difficult to spot this by analysing only the equations, since the equations deal with macro effects and the adjustment suggested to the equations in section 5.1.1 relates to the behaviour of individual agents, which then impacts the macro-level outcome.

This finding demonstrates the value of the understanding gained merely by writing an agent-based model, let alone running it. It was also revealing to see that the simple introduction of more than two schools caused the working-class preference to suddenly take on significance in the class sensitive case (see section 6.3); in the two-school model investigated by Room and Britton, it has no effect. Thus, our understanding of the system's dynamics was deepened further.

Running the model also generated some interesting results; the key one being, as discussed in section 7.2, that the system has a built-in inequity defined by the school's strategies and the children's preferences, which cannot be permanently influenced without changing these intrinsic parameters.

The expectation that it would be relatively easy to build a model that would generate rich results was confirmed; and it would not be difficult to add extensions to it that would produce more fruitful output yet. In contrast, describing the interactions seen here in equations would be almost impossible, and would certainly be a very difficult and error-prone task.

However, the project suffered under the model's poor performance, which caused the entire schedule to be shifted forward to await the results of the final experiments. It was impossible to predict at the planning stage how long an experiment would take to run; and in analysing the results, additional experiments were often discovered to be of interest, and were therefore added to the list, further increasing the delay. However, it would perhaps have been better to add a greater 'buffer' to the project plan to account for this type of unpredictable delay. Nevertheless, the project's evolutionary approach of developing the model and running experiments in parallel served it well; if a more traditional methodology had been chosen, it would have been impossible to run sufficient experiments in time. Moreover, the experiments served as tests of the model, and would have had to be run to some degree during development in any case.

8.2 Choice of technology

Overall, it seems that appropriate technology was chosen for the project, despite the performance issues introduced by Drools. Once the new version of Drools is released, it should be possible to improve on the simulation's performance significantly. Alternatively, the rules could be converted to JESS and profiling undertaken to see if a performance gain could be seen; but it seems most sensible to first assess the performance benefits brought by the new version of Drools.

Repast was a very 'friendly' platform to use: there was plenty of documentation available online, and most problems encountered had already been discussed on the mailing list. It was also very stable; no Repast bugs were encountered. The only drawback was, as discussed in section 3.3.2, that the library classes could not be trusted to preserve replicability of experiments without first reading their source code. Repast was indeed very flexible as anticipated, and all of the extensions discussed below should be easy to implement in Repast. It is, however, unknown how suitable it would be for a distributed system, although no obvious problems are yet apparent. The design of using one Drools working memory per agent certainly lends itself very well to a distributed model; indeed, this was in part the motivation for adopting the design.

8.3 Further work

8.3.1 Catchment areas

The idea of catchment areas follows on naturally from the consideration of more than two schools in a simulation. Children living in a particular school's catchment area would be given precedence over children living outside the area. This would lead to house prices rising in the area surrounding a popular school, pricing working-class parents out of the market in this more 'desirable' area, thus giving schools an indirect mechanism to cherry-pick middle-class students.

This effect is, however, already one of the motivations for assigning middle-class parents a higher preference for high-performing schools than working-class parents. This 'preference' is in part characteristic of middle-class parents' greater willingness, but also their greater *ability* to move to the most popular catchment areas. (While the term 'preference' is perhaps not quite the correct term, it is convenient to use in this context.)

Catchment areas were not implemented in this model, because they are so domain-specific. Even though the model does aim to capture the essence of the dynamics of the school admissions process, it was also important not to fall into Gilbert and Doran's "trap of verisimilitude" (2005, p.12). It was felt that implementing something as domain-specific as catchment areas would be straying too far from the project's motivation of exploring the usefulness of agent-based modelling techniques applied to this type of complex system, compared to the original equation-based model. This could make it more difficult to adapt the basic model to other, similar domains, and would arguably not enable new types of insight into the value of the model. If the model were to be further specialised in the domain of school choice, catchment areas would certainly be a useful addition, but otherwise, they might only add unnecessary complexity. The same argument can be applied to the suggestion of families moving house during the academic year, and being compelled to join whichever school happens to have a free place.

8.3.2 Successful schools expanding, 'failing' schools closing down

Another feature that could be added to the simulation would be to re-distribute places from a failing school to a more successful one. This eventuality is already provided for by the Java code; it would only require rules to be added to increase or decrease the number of places in a school. Inside a school's own expert system is perhaps not the best place for such rules, since a failing school would hardly be expected to offer to gradually close down, but adding a 'central authority' responsible for imposing such punishments and rewards could be seen as adding unnecessary overheads. It could be argued that it is not inappropriate to place such rules inside a school's expert system after all since they represent a constraint on its behaviour that a school would be aware of; and certainly, it would be a very cheap way to investigate this new aspect of the model.

It would be interesting to see whether a successful school expanding would actually negatively affects its success, since it might need to accept more working-class applicants to fill those new places.

8.3.3 More intelligent agents

The child agents in particular are very primitive, and would benefit from increased intelligence. For example, a social influence mechanism could be added to pass on interest in the specialist subjects in a local community. For example, an area could be especially sports-oriented if it boasted a well-equipped sports centre, or sought after by linguists if there is a specialist language primary school in the vicinity. This would also allow the school specialisms/child interests levels to become more truly co-evolutionary: at the moment, the schools have no motivation to adapt to any skew in the children's interest levels since they will be reset (without using any historic trends) in the next academic year. The schools therefore base their choice of specialism purely on the specialisms of the other schools. The children do not evolve at all – although they do shape their environment (the schools) for their successors. Since each child only has a one-year lifetime, it seems that the only way they could properly be said to co-evolve is if they also influenced their successors more directly. Their lifespan of interest could also be increased, giving them social influence throughout their time at school; perhaps their own preferences would also be affected dynamically by the dominant interests of their classmates during this time, and by the specialism of the school.

On a more basic level, the child rules could also be adapted to include some of the strategies the author thought of while analysing the experimental results. If working-class children knew that they were undesirable to a class sensitive school, they could assume in a multi-school scenario that they would not be accepted to their first-choice school and instead apply directly to their second-choice school in the first application round. All of the middle-class children would probably have applied to the leading school in the first round, so there would be little competition for the second-best school, even if it was excellent. However, middle-class children might anticipate this, and, say, 15% of them, the most risk-averse ones, would apply to the second-best school in the second application round, to avoid the risk of being left only with the third-rated school due to insufficient places in the others.

This would mean using different rule sets for working- and middle-class children.

Similarly, rules might be added for children to anticipate which schools are so popular that they would be wasted as a second choice. If there are only two excellent schools, it makes sense to list one as a first choice, and perhaps the third-best school as a second choice, since both excellent schools can be expected to fill up in the first application round. The child should not waste its second choice on a school that it knows will not have any places left.

Children might also differentiate between two schools having an equal league table position, with one holding a stable position and the other being perceived as 'on the slide'. Perhaps a highly rated school that was declining might even be passed over in favour of a less well-rated school that was improving, since the children will be attending the school for six years.

The school agents could also benefit from increased intelligence: for example, when choosing a new specialism, they simply pick the first one which meets their basic criteria, rather than evaluating all possible options and ranking them by suitability. The (unethical) 'act of sabotage' rule in which a school adopts a new specialism to destroy the niche position of a competitor, for example, could be made more effective by preferring a new specialism that destroyed the joint niche of *two* competitors at once.

More significantly, the school agents could also learn from their mistakes by monitoring the effects of a strategy change. For example, if a school considers itself popular enough to become class sensitive but the children's class blind preferences turn out to be so high that this move backfires and it starts slipping down the league tables, it could recognise the situation and revert to its previous strategy, remembering not to make the same error again. This would make the rules much more complex, but it would also stop 'stupid' runs from occurring where the schools take turns using the same bad strategy and switching roles, when any human could recognise the pattern and advise them to stop (e.g. Figure 6.13). An even more intelligent school could even learn from its competitors' mistakes in this manner.

8.3.4 Increased heterogeneity among agents

Currently, there are only three distinct types of agent within a given simulation run: working-class children, middle-class children, and schools. It is true that these groups act differently when co-evolutionary behaviour is enabled, but even then, each group's members follow homogeneous rules. They only differ in their attributes. Arguably, the middle- and working-class children differ only in attributes as well, so there are only two types of agent. However, this still produces heterogeneous behaviour, since different attributes lead to different rules being triggered.

It would, however, be more interesting to sub-divide these groups, allocating, say, 10% of school agents a more aggressively competitive set of rules, and giving the remainder a more cooperative rule set. This could easily be done in Repast by defining a new subclass of `School`, in the same way that `EvolutionarySchool` was created and given its own rule set, and adding a simulation parameter that specified what percentage of agents should be of this type. This is already done via the FractionMiddleClassPopulation parameter for creating that percentage of middle-class child agents, making the remainder working-class.

8.3.5 More sophisticated league table calculations

The league table ratings used in this project were kept relatively abstract throughout: they only consisted of a four-year moving average of the school's middle-class composition. It could be interesting to examine more sophisticated league table mechanisms, such as the value-added tables used in real life (DfES, 2007). This could be calculated by allocating each child an 'exam score' on school entry that was on average lower for working-class children. Upon leaving the school, they would be given a 'value-added' exam score that was dependent both on their original exam score and on the social make-up of the school, since a child's exam results are likely to be positively affected by a high percentage of middle-class children in its class (Lauder and Hughes, 1999).

An alternative view of league tables would be to abstract them even further, and instead of using raw ratings, to provide parents with only ranking information. This would probably significantly influence the simulation's outcome, perhaps preventing any one school from gaining a fully polarised intake since its rising degree of success would be masked from view. Middle-class parents should view the difference between the top and second-highest ranked schools as larger than working-class parents do, analogous to middle-class parents' current increased preference for high league table ratings.

8.3.6 More flexible model parameters

A trade-off has been present throughout the project between not wishing to over-complicate the model's parameters, and allowing for sufficient flexibility. Currently, some of the necessary flexibility requires the user to use a different random seed on occasion to obtain the precise starting configuration desired. This would clearly be unacceptable in a more rigorous research project, since it would compromise the validity of comparing simulation runs against each other directly. A useful extension would therefore be to write a custom parameter file reader that would overcome the constraints on expressiveness imposed by Repast.

The constraints encountered by this project all followed the same format: we had a parameter describing a variable number of objects or events, such as the number of possible school specialisms, or the number of stochastic shocks; we then wished to further parameterise each of these objects, one by one. Repast does not allow a variable number of parameters, the number of which depends on another parameter. This could, however, easily be overcome by writing a custom parameter file, since Repast allows the desired reader to be specified within a parameter file; but this would have the disadvantage of no longer being compatible with Repast's parameter GUI, so Repast could no longer be used to generate real-time graphs in its interactive mode.

Since Repast is open-source, existing functionality could easily be adapted for the parameter file reader, and only this particular extra feature would need to be written. It was not implemented on this project due to time constraints, and because it only became a necessity at the end of the project, when using school specialisms; until then, we had been lucky and had not needed to use an alternative random seed. We were, however, not able to reliably compare the three-specialism cases in those experiments against the rest of the experiments.

Another implementation adjustment that would be useful would be to turn the rule files used into parameters to the program. This would allow the simulation itself to be packaged as a JAR file, along with an appropriate script to run it, and the rules could be kept separately as text files. This would allow non-technical social scientists to edit the Drools rules, which are very human-readable (see Appendix C, sections 11.1-11.4), without having to compile or even see any Java code. It would also allow long batch runs to be created that used different rules for different runs. In this project, in order to run experiments on several different rule sets, such batch runs were separated into several sub-runs, one per set of rules.

8.4 Predictive abilities

The model does of course not even approach the level of realism required to make a prediction about the system it simulates with confidence. In fact, the system it models is arguably really that described by Room and Britton's equations, rather than the real world. We have aimed more towards a "thought experiment" type of model than one aimed towards prediction: as noted in section 2.4, the two may be mutually exclusive. And even if it were a validated replica of the real world school admission domain,

prediction in the social sciences is notoriously poor, as discussed in section 2.4. However, it could nonetheless be used to make a generalised, abstract prediction about a real world scenario, with the caveat that its reliability is unknown.

Brighton's school admissions process has recently received much media attention (e.g. BBC News, 2007; The Observer, 2007). They are replacing the old system, which was based on the distance of a child's home to the school, by a "lottery" system of random allocation that is equivalent to our 'class blind' school strategy. The changes are motivated by new government guidelines designed to reduce social exclusion by preventing wealthy families moving into the most popular catchment areas, pricing working-class families out of the market.

We do not model catchment areas explicitly, but this is one of the factors that supports the higher preference for high-performing schools held by middle-class children, compared to working-class children. Our scenario of two class sensitive schools leading towards social polarisation of school composition therefore corresponds approximately to the original Brighton configuration.

Our model does not enable us to force schools to change strategy at a particular point in time – the schools decide themselves if and when they wish to do so – but we can specify an initial configuration involving inequity. We could therefore 'fast-forward' to the current point in time and run a simulation with an inequity of, say, 80% between two schools[32] and tell the schools to use class blind allocation policies. If co-evolutionary behaviour is disabled, then they are forced to remain in the Brighton scenario. In fact, we have already run this simulation, in experiment 6.4.10. It predicts that the inequity will decrease until it reaches the intrinsic inequity defined by the new school strategies. The magnitude of this new inequity cannot be predicted, since we lack the empirical data to decide upon appropriate middle- and working-class preferences for high-performing schools.

In short, we predict that the inequity between the schools will be lessened, but only up to a point, after which it will stabilise at the level built in to the new scenario. The government will only succeed partially in its aim to reduce social exclusion, since it can only fully redefine the schools' behaviour through social policy; it only has indirect influence on families' preferences. However, it could be that middle-class parental preferences are also reduced because of the policy changes, since they lose their advantage of being able to move to a popular catchment area. This would reduce the resultant inequity further yet. But their preferences would still not be reduced to the level of working-class parents' preferences, eliminating the inequity entirely, since there are many other factors that shape their preferences (Room and Britton, 2006a).

As long as the class system exists in its current form, there will be social inequity; the government can only act to minimise the social exclusion as far as possible.

[32] Unfortunately the model does not allow initial inequity to be specified between more than two schools.

Bibliography

Albin, P. S. (1998). Barriers and Bounds of Rationality. Princeton: Princeton University Press.

Axelrod, R. (1997). Advancing the art of simulation in the social sciences. *Complexity* **3**(2) pp. 16-22

Axtell, R. (1999). Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences. In *Workshop on Agent Simulation: Applications, Models, and Tools* Macal, C. M. and Sallach, D. (Eds.). Chicago: Social Science Research Computation, University of Chicago pp. 3-24

Balci, O. (2003). Verification, Validation, and Certification of Modeling and Simulation Applications. **In**: *Proceedings of the 2003 Winter Simulation Conference* (Chick, S., Sanchez, P. J., Ferrin, D., Morrice, D. J., ed.). New Orleans: Winter Simulation Conference.

Batten, D. F. (2000). Discovering Artificial Economics: How Agents Learn and Economies Evolve. New York: Westview Press

BBC News (2007). Schools to give places by lottery. Article of 28 February. [Online] available at http://news.bbc.co.uk/1/hi/education/6403017.stm (accessed 30 April 2007)

Birdsall, N. and Meesook, O. A. (1986). Children's Education and the Intergenerational Transmission of Inequality: a Simulation. *Economics of Education Review* **5**(3) pp.239-256

Board, R. (1994). Polynomially Bounded Rationality. *Journal of Economic Theory* **63**(2) pp. 246-270

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the USA (PNAS)* **99**(3) pp. 7280-

7287. [Online] available at http://www.pnas.org/cgi/content/full/99/suppl_3/7280 (accessed 29 November 2006)

Bowe, R., Gewirtz, S. and Ball, S. J. (1994). Captured by the Discourse? Issues and concerns in researching 'parental choice'. *British Journal of Sociology of Education* **15**(1) pp. 63-78

Brenner, T. and Werker, C. (2006). A Practical Guide to Inference in Simulation Models. Working Paper 0602, Papers on Economics & Evolution, Max Planck Institute of Economics, Evolutionary Economics Group, Jena, Germany [Online] available at http://www.econ.iastate.edu/tesfatsi/EmpValidACE.WerkerBrenner.0602.2006.pdf (accessed 2 December 2006)

Brueckner, S. A. and Parunak, H. V. D. (2003). Resource-Aware Exploration of the Emergent Dynamics of Simulated Systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (Melbourne, Australia, July 14 - 18, 2003). AAMAS '03. New York: ACM Press pp. 781-788

Bryson, J. J., Yasushi, A., and Lehmann, H. (2006). Agent-based models as scientific methodology: A case study analysing primate social behaviour. *Philosophical Transactions of the Royal Society, B*. In press.

Cederman, L.-E. (2005). Computational Models of Social Forms: Advancing Generative Process Theory. *American Journal of Sociology* **110**(4) pp. 864-893

Codehaus Foundation (2006). Rete. [Online] available at http://legacy.drools.codehaus.org/Rete (accessed 23 April 2007)

Conte, R., Edmonds, B. Moss, S. and Sawyer, R. K. (2001). Sociology and Social Theory in Agent Based Social Simulation: A Symposium. *Computational and Mathematical Organization Theory* **7** pp. 183-205

Conte, R., Gilbert, N. and Sichman, J. S. (1998). MAS and Social Simulation: A Suitable

Commitment. In *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation (MABS)*. Sichman, J. S., Conte, R. and Gilbert, N. (Eds.). London: Springer-Verlag pp. 1-9

David, N., Sichman, J. and Coelho, H. (2002). Towards an Emergence-Driven Software Process for Agent-Based Simulation. In *Proceedings of Multi-Agent-Based Simulation II: Third International Workshop, MABS 2002, Bologna, Italy, July 15-16, 2002* pp. 89-104. Coelho, H., Sichman, J., Drogoul, A. and David, N. (Eds.). Berlin: Springer-Verlag

David, N., Sichman, J. and Coelho, H. (2005). The Logic of the Method of Agent-Based Simulation in the Social Science: Empirical and Intentional Adequacy of Computer Programs. *Journal of Artificial Societies and Social Simulation* **8**(4). [Online] available at http://jasss.soc.surrey.ac.uk/8/4/2.html (accessed 2 December 2006)

Davidsson, P. (2000). Multi Agent Based Simulation: Beyond Social Simulation. In *Proceedings of the second international workshop on Multi-agent based simulation (MABS)*. Moss, S. and Edmonds, B. (Eds.). London: Springer-Verlag pp. 97-107

Davies, N. (1999). Bias that killed the dream of equality. *Guardian Unlimited*, September 15. [Online] available at http://education.guardian.co.uk/specialreports/educationincrisis/story/0,,84125,00.html (accessed 22 October 2006)

DfES (2003a). Specialist Schools Programme 2003/2004: Applications – A Guide for Schools. Ref: DfES/0366/2003. [Online] available at http://publications.teachernet.gov.uk/eOrderingDownload/DfES%200366%20200MIG2256.pdf (accessed 22 April 2007)

DfES (2003b). School Admissions Code of Practice. Ref: DfES/0031/2003. [Online] available at http://publications.teachernet.gov.uk/eOrderingDownload/DfES%200031%20200MIG2076.pdf (accessed 22 April 2007)

DfES (2007). 2006 Post-16 Pilot: Technical Annex. [Online] available at http://www.dfes.gov.uk/performancetables/pilotks5_06/a5.shtml (accessed 30 April 2007)

Dietrich, J. (2003). The Mandarax 3.0 Manual. [Online] available at http://mandarax.sourceforge.net/docs/mandarax.pdf (accessed 23 April 2007)

Dubiel, B. and Tsimhoni, O. (2005). Integrating agent based modeling into a discrete event simulation. In *Proceedings of the 37th Conference on Winter Simulation* (Orlando, Florida, December 04 - 07, 2005)

Edmonds, B. and Hales, D. (2003). Replication, Replication and Replication – Some Hard Lessons from Model Alignment. *Journal of Artificial Societies and Social Simulation* **6**(4). [Online] available at http://jasss.soc.surrey.ac.uk/6/4/11.html (accessed 10 December 2006)

Edwards, M., Huet, S., Goreaud, F. and Deffuant, G. (2003). Comparing individual-based model of behaviour diffusion with its mean field aggregated approximation. *Journal of Artificial Societies and Social Simulation* **6**(4). [Online] available at http://jasss.soc.surrey.ac.uk/6/4/9.html (accessed 2 December 2006)

Epstein, J. M. and Axtell, R. (1996). Growing Artificial Societies: Social Science From The Bottom Up. Cambridge: MIT Press.

Flake, G. W. (1998). The Computational Beauty of Nature. Cambridge, Massachusetts: MIT Press.

Fowler. J. H. and Smirnov, O. (2005). Dynamic Parties and Social Turnout: An Agent-Based Model. *American Journal of Sociology* **110**(4) pp. 1070-1094

Geer, D. (2005). Eclipse becomes the dominant Java IDE. *Computer* **38**(7), pp. 17-18.

Gilbert, N. (1995). Simulation: an emergent perspective. The text of a lecture first given at the

conference on New Technologies in the Social Sciences, 27-29th October 1995, Bournemouth, UK. [Online] available at http://www.soc.surrey.ac.uk/research/cress/resources/emergent.html (accessed 24 November 2006)

Gilbert, N. (2004). Agent-based social simulation: dealing with complexity. [Online] available at http://www.soc.surrey.ac.uk/NG_pubs/paper165_NG.pdf (accessed 29 November 2006)

Gilbert, N. and Conte, R. (Eds.) (1995). Artificial societies: the computer simulation of social life. London: UCL Press

Gilbert, N. and Doran. J. (Eds.) (1994). Simulating societies: The computer simulation of social phenomena. London: UCL Press.

Gilbert, N. and Terna, P. (2000). How to build and use agent-based models in social science. *Mind and Society* **1**. pp. 57-72.

Gilbert, N. and Troitzsch, K. G. (1999). Simulation for the Social Scientist. Buckingham: Open University Press

Gulyás, L. (2005). Understanding Emergent Social Phenomena. Thesis (Ph.D.). Loránd Eötvös University, Budapest, Hungary.

Hales, D., Rouchier, J. and Edmonds, B. (2003). Model-to-Model Analysis. *Journal of Artificial Societies and Social Simulation* **6**(4). [Online] available at http://jasss.soc.surrey.ac.uk/6/4/5.html (accessed 1 December 2006)

Hegselmann, R. and Flache, A. (1998). Understanding Complex Social Dynamics: A Plea For Cellular Automata Based Modelling. *Journal of Artificial Societies and Social Simulation* **1**(3). [Online] available at http://www.soc.surrey.ac.uk/JASSS/1/3/1.html (accessed 3 December 2006)

Helleboogh, A., Vizzari, G., Uhrmacher, A. and Michel, F. (2007). Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems* **14**(1)

Henshel, R. L. (1982). Sociology and Social Forecasting. *Annual Review of Sociology* **8** pp. 57-79

Hoyle, R. B. and Robinson, J. C. (2003). League tables and school effectiveness: a mathematical model. *Proceedings of the Royal Society B: Biological Sciences* **270**(1511) pp. 113-119

Huhns, M. N. and Singh, M. P. (Eds.) (1998). Readings In Agents. San Francisco: Morgan Kaufmann.

Janzen, D. H. (1980). When is it Coevolution? *Evolution* **34**(3) pp. 611-612.

JBoss (2007). 3.1M1ReleaseNotes. [Online] available at http://wiki.jboss.org/wiki/Wiki.jsp?page=3.1M1ReleaseNotes (accessed 23 April 2007)

Jennings, N. R., Sycara, K. and Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems* **1** pp. 7-38

Jordan, B. (1996). A Theory of Poverty and Social Exclusion. Cambridge: Polity Press.

Karlsson, G. (1969). Simulation Models and the Problem of Relating Social Systems to Individual Behaviour. *Quality and Quantity* **3**(1-2) pp. 168-175

Karsten, S., Visscher, A., and De Jong, T. (2001). Another Side to the Coin: the unintended effects of the publication of school performance data in England and France. *Comparative Education* **37**(2) pp. 231-242.

Kleijnen, J. P. C. (1995). Sensitivity Analysis and Related Analyses: a Survey of Statistical Techniques. Paper presented at the *International Symposium SAMO95 (Theory and Applications of Sensitivity Analysis of Model Output in Computer Simulation)*, 25–27 September 1995, Belgirate, Italy.

Küppers, G. and Lenhard, J. (2005). Validation of Simulation: Patterns in the Social and

Natural Sciences. *Journal of Artificial Societies and Social Simulation* **8**(4) [Online] available at http://jasss.soc.surrey.ac.uk/8/4/3.html (accessed 10 December 2006)

Latané, B. (1996). Dynamic Social Impact: The Creation of Culture by Communication. *Journal of Communication* **46**(4) pp.13-25

Lauder, H., Hughes, D. (1999). Trading in Futures: Why Markets in Education Don't Work. Milton Keynes: Open University Press

Luck, M., McBurney, P. and Preist, C. (2004). A Manifesto for Agent Technology: Towards Next Generation Computing. *Autonomous Agents and Multi-Agent Systems* **9** pp. 203-252

Macy, M. W. and Willer, R. (2002). From Factors to Actors: Computational Sociology and Agent-Based Modeling. *Annual Review of Sociology* **28** pp. 143-166.

Makowsky, M. (2006). An Agent-Based Model of Mortality Shocks, Intergenerational Effects, and Urban Crime. *Journal of Artificial Societies and Social Simulation* **9**(2). [Online] available at http://jasss.soc.surrey.ac.uk/9/2/7.html (accessed 28 November 2006)

Manski, C. F. (1992). Educational Choice (Vouchers) and Social Mobility. *Economics of Education Review* **11**(4) pp. 351-369

McFadzean, D. and Tesfatsion, L. (1999). A C++ Platform for the Evolution of Trade Networks. *Computational Economics* **14**(1-2) pp. 109-134

McIntyre, L. (1998). Complexity: A philosopher's reflections. *Complexity* **3**(6) pp. 26-32.

Meraviglia, C. (1996). Models of representation of social mobility and inequality systems. A neural network approach. *Quality and Quantity* **30** pp. 231-252

Miller, J. H. (1998). Active Nonlinear Tests (ANTs) of Complex Simulation Models. *Management Science* **44**(6) pp.820-830

Mitleton-Kelly, E. (1997). Organisations as co-evolving complex adaptive systems. Business Process Resource Centre Paper No.5, presented at British Academy of Management Conference, 8-10 September 1997, London (unpublished).

Mosler, H.-J. (2000). Computersimulation sozialpsychologischer Theorien. Weinheim: BELTZ PsychologieVerlagsUnion. [In German]

Mosler, H.-J. (2004). Modelling social behaviour with a socio psychological simulation approach. *Web Intelligence and Agent* **2** pp. 185-200

Moss, S. and Edmonds, B. (2005a). Sociology and Simulation: Statistical and Qualitative Cross-Validation. *American Journal of Sociology* **110**(4) pp. 1095-1131

Moss, S. and Edmonds, B. (2005b). Towards Good Social Science. *Journal of Artificial Societies and Social Simulation* **8**(4). [Online] available at http://jasss.soc.surrey.ac.uk/8/4/13.html (accessed 10 December 2006)

Odell, J., Parunak, H., Fleischer, M. and Brueckner, S. (2003). Modeling Agents and their Environment. In Mylopoulos, J., Winikoff, M., and Jennings, N. R. (Eds) *Agent-Oriented Software Engineering III: Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002. Revised Papers and Invited Contributions*. Berlin: Springer-Verlag, pp. 16-31

Office for National Statistics (2004). NS-SEC Classes and Collapses. [Online] available at http://www.statistics.gov.uk/methods_quality/ns_sec/class_collapse.asp (accessed 19 April 2007)

Ostrom, T.M. (1988). Computer Simulation: The Third Symbol System. *Journal of Experimental Social Psychology* **24**(5) pp. 381-392

Parunak, H. V. D., Savit, R. and Riolo, R. (1998). Agent-based Modelling vs. Equation-Based Modelling: A Case Study and Users' Guide. **In**: *Multi-Agent Systems and Agent-Based Simulation First International Workshop, MABS '98* (Sichman, J. S., Conte, R., and Gilbert, N.,

ed.). Berlin: Springer Verlag.

Plank, D. N. and Sykes, G. (1999). How choice changes the education system: A Michigan case study. *International Review of Education* **45**(5/6) pp. 385-416

Proctor, M., Neale, M., Lin, P. and Frandsen, M. (2006). Drools Documentation (version 3.0.6). [Online] available at http://labs.jboss.com/file-access/default/members/jbossrules/freezone/docs/3.0.5/html/index.html (accessed 23 April 2007)

Room, G. (1999). Social exclusion, solidarity and the challenge of globalisation. *International Journal of Social Welfare* **8**(3) pp. 166-174.

Room, G. and Britton, N. (2006a). The dynamics of social exclusion. *International Journal of Social Welfare* **15**(4) pp. 280-289.

Room, G. and Britton, N. (2006b). The dynamics of social exclusion – Mathematical Annex. [Online] available at http://people.bath.ac.uk/masnfb/schools.pdf (accessed 22 October 2006)

Sawyer, R. K. (2003). Artificial Societies: Multiagent Systems and the Micro-Macro Link in Sociological Theory. *Sociological Methods Research* **31**(3) pp. 325-363

Schelling, T.C. (1978). Micromotives and Macrobehavior. New York: W. W. Norton & Company.

Scott, J. and Marshall, G. (eds) (2005). Oxford Dictionary of Sociology. Oxford: Oxford University Press

Simmons, R. (Jr.). (2004). Hardcore Java. Cambridge, USA: O'Reilly Media

Tesfatsion, L. (2003). Agent-Based Computational Economics. ISU Economics Working Paper No. 1, Iowa State University. [Online] available at http://www.econ.iastate.edu/tesfatsi/acewp1.pdf (accessed 2 December 2006)

The Observer (2007). Give children a chance, not a lottery. Article of 4 March. [Online] available at http://education.guardian.co.uk/admissions/story/0,,2026157,00.html (accessed 30 April 2007)

Tobias, R. and Hofmann, C. (2004). Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation* **7**(1). [Online] available at http://jasss.soc.surrey.ac.uk/7/1/6.html (accessed 10 December 2006)

Vicsek. T. (2002). Complexity: The bigger picture. *Nature* **418** p. 131

Vidgen, R. and Wang, X. (2004). Adaptive Information System Development. In: Grant, K., Edgar, D.A., Jordan, M. (eds). Proceedings of the 9th UK Association for Information Systems Conference, 5-7 March 2004, Glasgow.

West, A. and Pennell, H. (2002). How new is New Labour? The quasi-market and English schools 1997 to 2001. *British Journal of Educational Studies* **50**(2) pp. 206-224.

Wooldridge, M. J. (2002). An Introduction to Multiagent Systems. Chichester: John Wiley & Sons

Wooldridge, M., Jennings, N. R. and Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems* **3** pp. 285-312

Yair, G. (1996). School Organization and Market Ecology: a realist sociological look at the infrastructure of school choice. *British Journal of Education* **17**(4) pp. 453-471

Zambonelli, F. and Omicini, A. (2004). Challenges and Research Directions in Agent-Oriented Software Engineering. *Autonomous Agents and Multi-Agent Systems* **9** pp. 253-283

9 Appendix A

9.1 Validation of initial prototype against Room and Britton's model

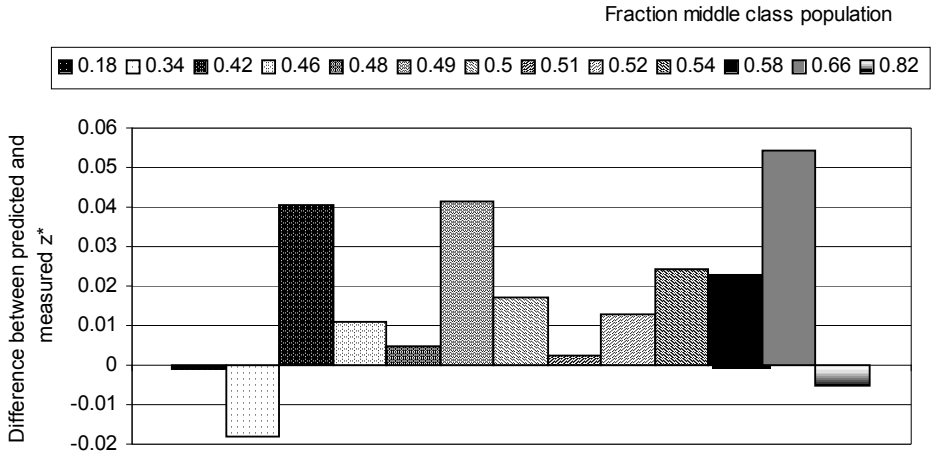


Figure 9.1 – Difference between z* (the inequity between the schools' middle-class intakes) predicted by Room and Britton's model, and that measured using the simulation, for varying middle-class fractions of the population. The differences appear to be acceptably low; we can conclude the model performs as expected. The runs used a middle-class preference of 2.56 and a working-class preference of 0.01.

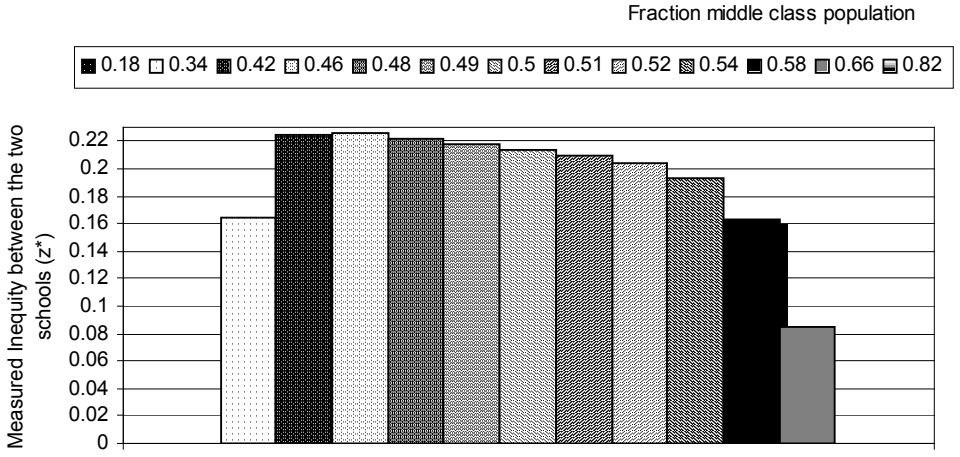


Figure 9.2 – Predicted inequity produced by varying the fraction of the population that is middle-class, according to Room and Britton's model. As previously, the intervals between bars are logarithmically spaced, centred around 0.5. The runs used a middle-class preference of 2.56 and a working-class preference of 0.01.

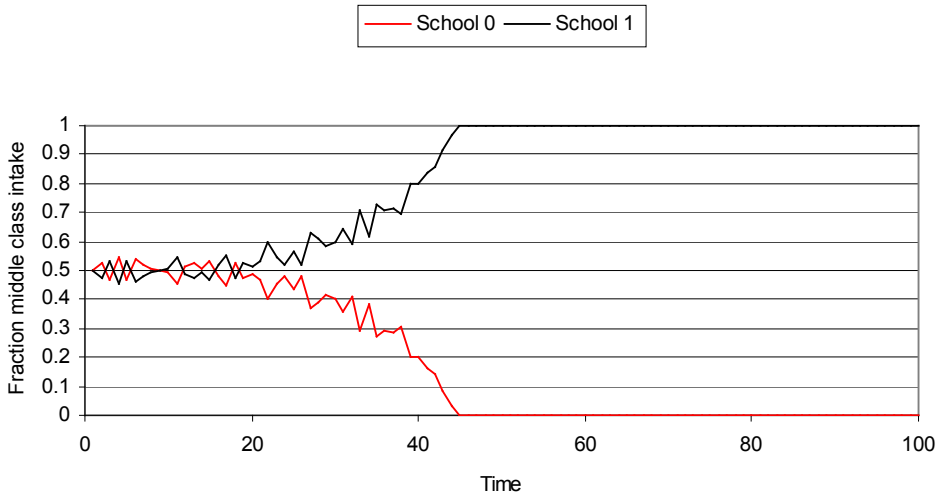


Figure 9.3 – Two class sensitive schools: polarisation is observed much sooner with a relatively high middle-class preference of 1.64 (the working-class preference is 0.82). Note that this figure uses a smaller scale than its predecessors.

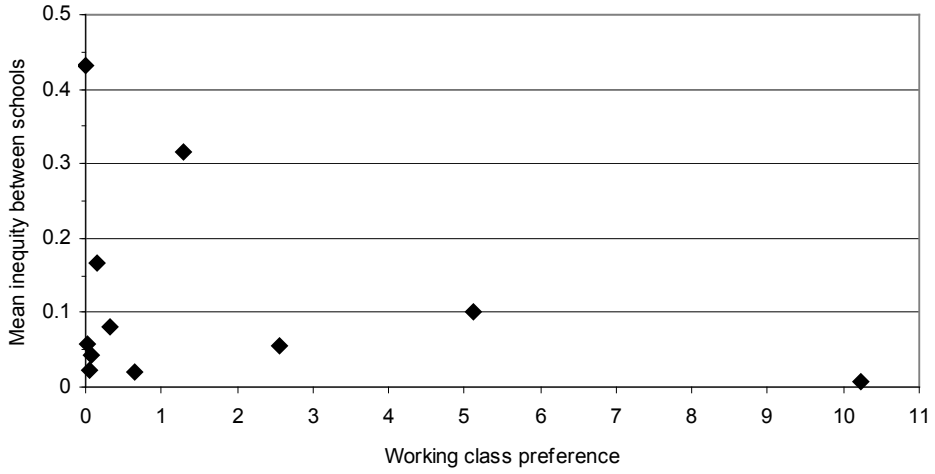


Figure 9.4 – Scatterplot of working-class preference against the mean inequity between the two schools, for middle-class preference 0.99. No relationship between the variables is found, as expected.

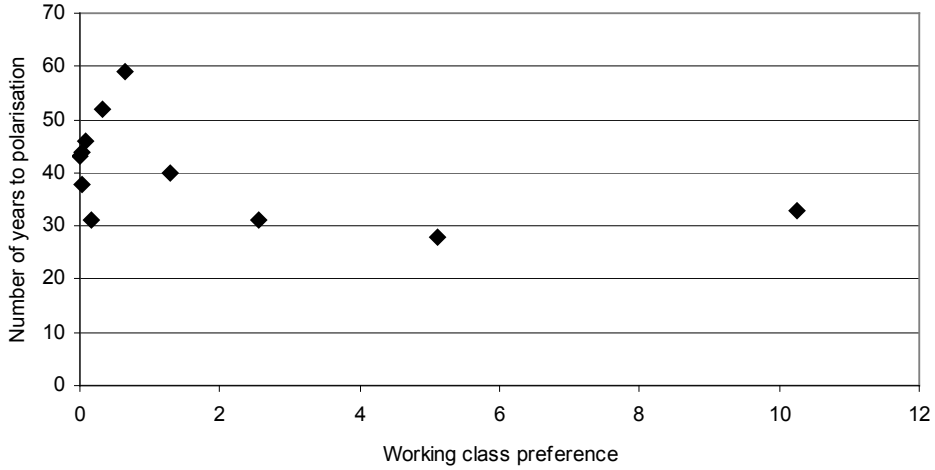


Figure 9.5 – Scatterplot of working-class preference versus number of years taken to reach total polarisation. No relationship can be identified.

9.2 Stochastic shocks

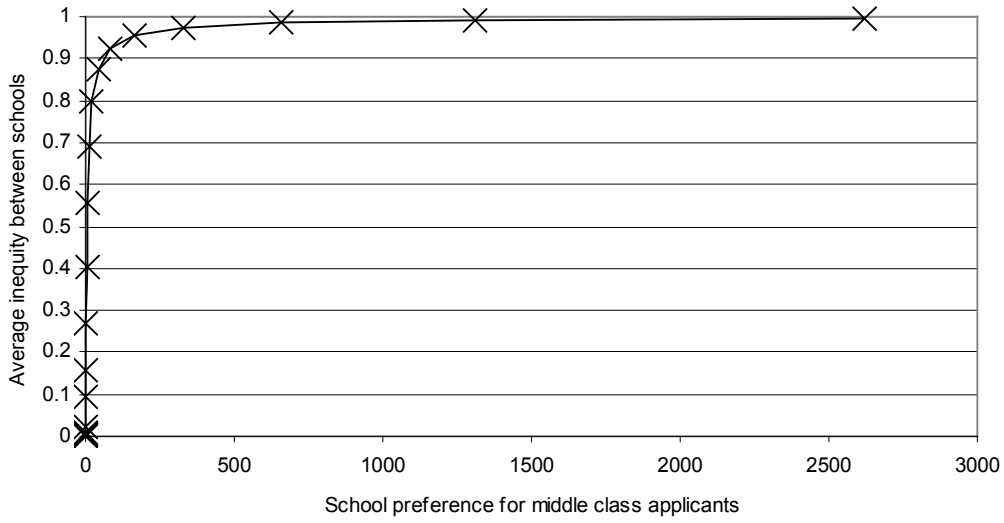


Figure 9.6 – The average inequity between schools increases exponentially with the schools' preference for middle-class applicants. Each run used an equally high middle- and working-class preference for middle-class schools of 20.48, and ran for 200 years using the same random seed. The first 5 years were excluded to allow the system to stabilise first.

9.3 Validating the Drools model

A number of tests were undertaken to validate the Drools model against the original Java model, to establish that it produces the same behaviour under the same conditions. The two models are far too different to be able compare two runs with the same parameters

and random seed and expect an identical result from each model, but the similarity between runs with the same parameters should be approximately that experienced when varying the random seed within the original prototype.

Unfortunately, since the Drools model takes much longer to run than the Java model (see section 3.3.5), it was not possible to run every single test run on the Java model in the Drools model as well. Additionally, the evolutionary nature of the project meant that development was ongoing up until the end, and the regression tests had to be re-run every time the Drools model changed. Instead, a cross-section of representative experiments had to be chosen for replication. These are summarised here. It should, however, be noted that the lack of complete tests means that potentially, there could be a section of the parameter space left untouched which exhibits anomalous behaviour.

9.3.1 Class blind schools

Middle-class and working-class preference parameter sweep

Due to time constraints, each run was only simulated for 200 years rather than the 500 the Java model used, so the Java model's results had to be recalculated using only 200 years to allow for an accurate comparison. We now show the differences between the Java model's and the Drools model's average inequity for each run:

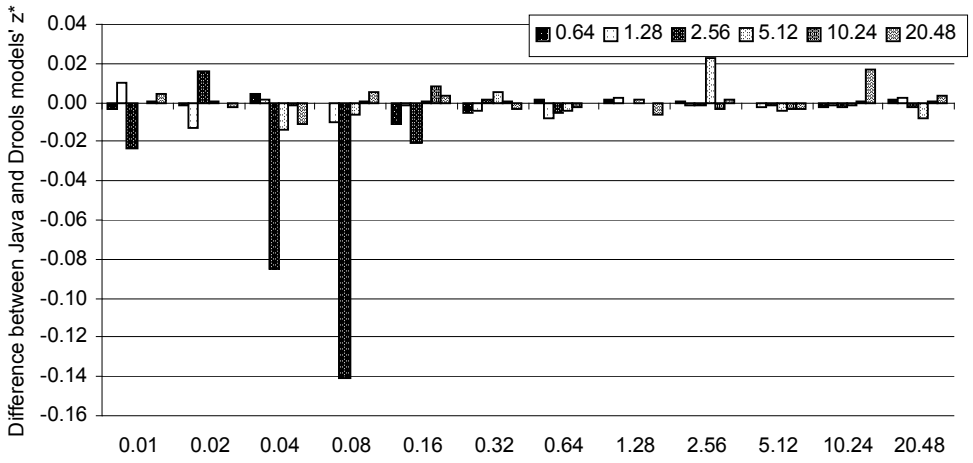


Figure 9.7 – Differences in average inequity (z*) between the Java and the Drools models' results for a parameter sweep of the middle- and working-class preferences. As in previous diagrams of this type, each column shading represents a different category of middle-class preference as detailed in the legend, while the categories of working-class preference are shown along the horizontal axis.

Figure 9.7 shows a close agreement in results, except for a middle-class preference of 2.56. Examining the individual runs soon revealed why this was the case – the one with the most severe difference is shown below for both models:

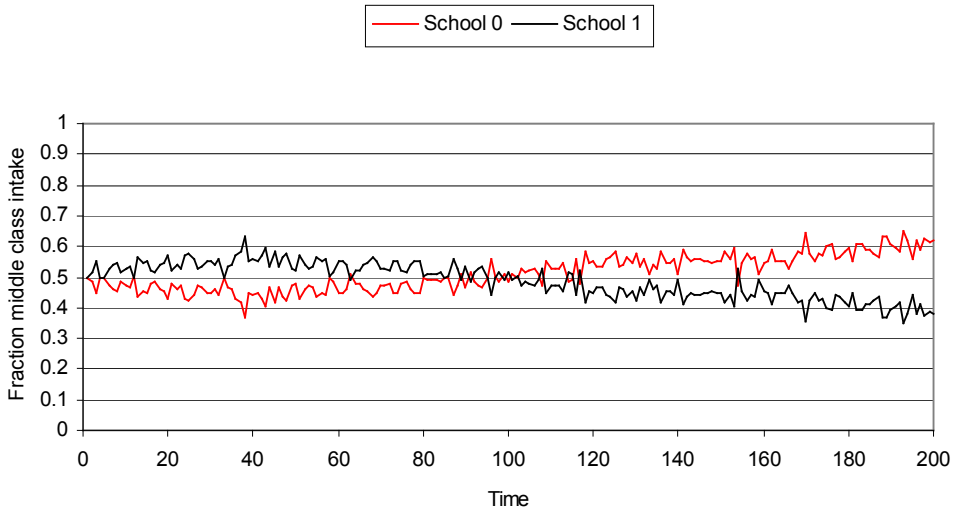


Figure 9.8 – Class blind schools with a middle-class parental preference of 2.56 and a working-class preference of 0.08, run in the Java model.

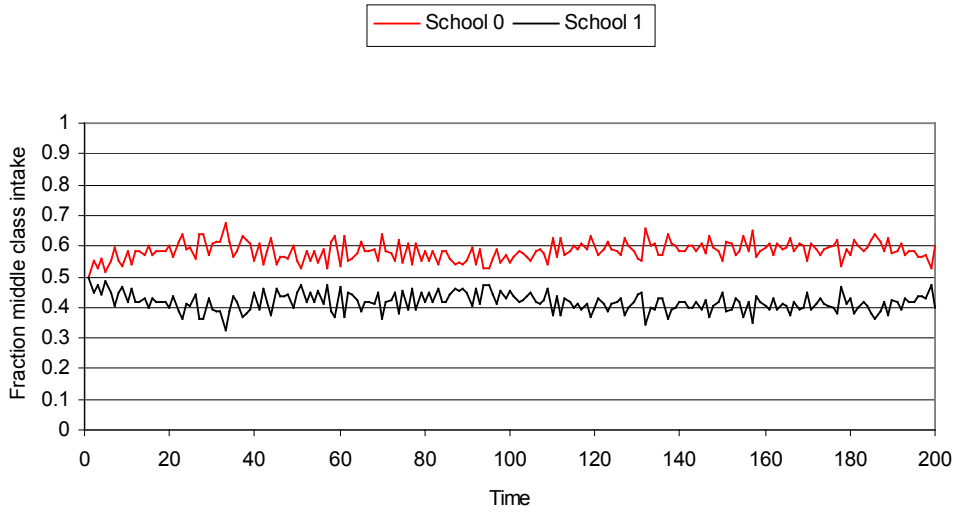


Figure 9.9 – Drools model version of Figure 9.8.

These figures indicate that the overall pattern is still the same, but coincidentally in this case the Java model experienced a role swap during the first 200 years, while the Drools model did not. The same thing happened in the other severe discrepancy, when the working-class preference is 0.04. In other runs of the Drools model using different random seeds, a crossover like that of Figure 9.8 was seen – it is a random occurrence. It seems that when the systems are this unstable, it is preferable to compare the overall pattern rather than try to produce an 'average inequity' where the direction of inequity is not fixed. With that perspective, it seems that the Drools model successfully replicates the Java model in this experiment.

To increase confidence in the Drools model, the Java model's in-depth experiment on middle-class preferences ranging from 1.25 to 2.45 was repeated as well, as shown below:

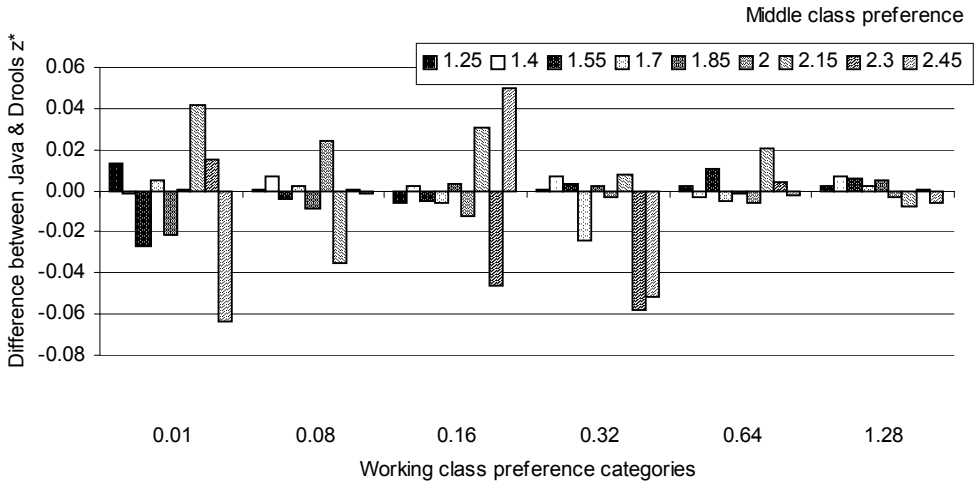


Figure 9.10 – Difference in the inequity between the two schools for the Java and the Drools models, under various combinations of middle- and working-class preferences. Each shade of bar represents a different middle-class preference (see legend), while the horizontal axis describes categories of working-class preference. (Note the different scale to Figure 9.7.)

A mean difference of 0.013 seems to be satisfactory, given that the standard deviation of a similar statistic between runs using different random seeds in the Java model was 0.011 for a middle-class preference of 2.56 (with a working-class preference of 0.08; see 5.1.3).

Sweep of middle-class fraction of the population

This was varied using a middle-class preference of 20.48, and a working-class preference of 0.01, since this case had previously produced very little variance in the Java model, meaning it should be easier to compare results.

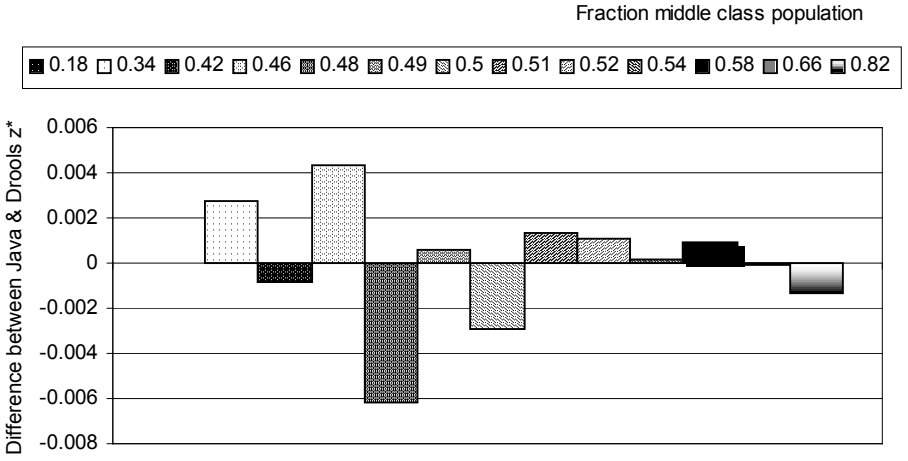


Figure 9.11 – Differences in inequity in the Java and the Drools models, for varying values of the fraction of the population that is middle-class, each of which is represented by a differently shaded bar. (Note that the intervals between bars are logarithmic, centred on 0.5, rather than being evenly spaced.)

The differences found here were very small compared to the inequities actually found (see Figure 5.8), and are much smaller than the differences between the Java model and

Room and Britton's model (see Figure 9.1) and so we can conclude that the models' behaviour coincides.

9.3.2 Class sensitive schools

Middle-class preference sweep

The same middle-class parameter as examined in the Java model was explored: preferences centred on the critical value of 1.0, increasing and decreasing logarithmically to get the sequence 0.84, 0.92, 0.96, 0.98, 0.99, 1, 1.01, 1.02, 1.04, 1.08, 1.16, 1.32, 1.64, 2.28, 3.56, 6.12 and 11.24. This resulted in a close match to the Java model's results:

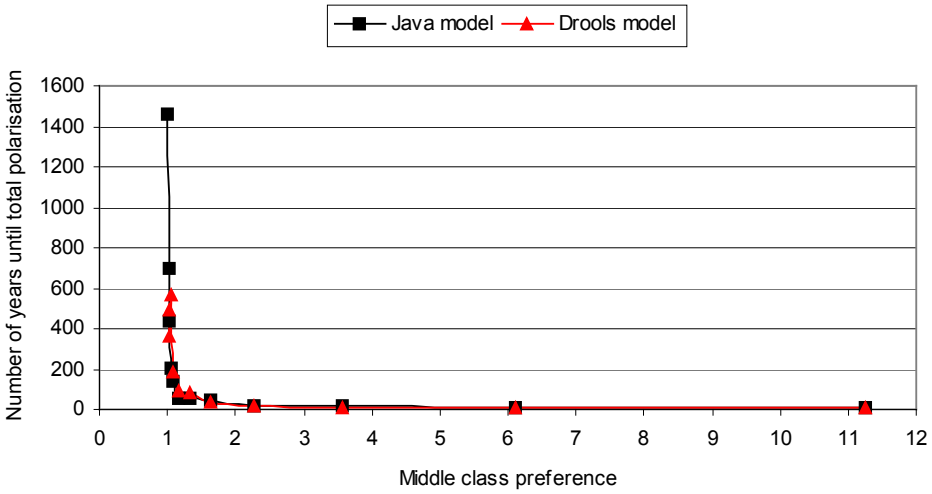


Figure 9.12 – Comparison between the Java and the Drools model of number of years until total polarisation occurred.

However, the Drools model is missing the first data point, for a middle-class preference of 1.0, since polarisation did not occur within the 1500 years tested with the usual random seed. Given that polarisation only occurred in year 1459 in the Java model, it is easy to imagine that polarisation could have been just around the corner in the Drools model – if it were not for the fact that the slow polarisation process began in the Java model around the year 600 (see Figure 5.12), and in the Drools model it subsides again:

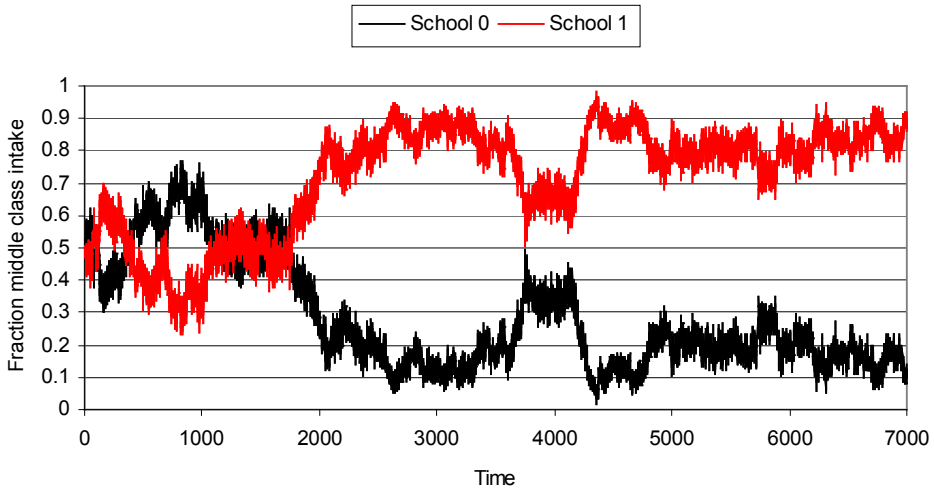


Figure 9.13 – Drools model: two class sensitive schools with a middle-class preference of 1.0, the critical value for a middle-class population of 50%. This simulation case was re-run to see if the point of polarisation could be identified, but we were forced to give up after 7000 years, due to lack of time. While the potential for polarisation is there, we could continue indefinitely and never hit it.

Nevertheless, it appears as though the potential for polarisation is still there in Figure 9.13, but in a very unstable sense. This is consistent with our assumption that we cannot predict whether polarisation will occur at the critical value, and so the discrepancy between the two models does not seem very serious.

What is more serious is that in the early cases immediately following the critical value, with a middle-class preference of 1.01 and 1.02, the Drools model reaches polarisation significantly sooner than the Java model. However, this concern is lessened in the light of the huge variation found within the Java model for a middle-class preference of 1.01, where results ranged from 369 to 1449 years (see section 5.2.2). No 'representative run' can be identified for the Java model in these borderline cases, and so these individual runs here cannot really be compared either. If the Drools model were also run with 100 different random seeds for a middle-class preference of 100, for 1500 years each, then a more reasonable comparison would be possible; unfortunately, there was no time for this in this project. All this experiment can tell us is that the results here are within the range covered by the Java model; it could, however, be that the Drools' models range overlaps with that of the Java model, but is not identical to it.

Middle-class/working-class preference sweep

The purpose of these tests was to confirm that, as before, the working-class preference has no significant impact on class sensitive results.

a) Middle-class preference of 0.99

As in the Java model, no polarisation between the schools' middle-class intakes occurred, yet very volatile behaviour was exhibited. In the Java model, a marked inequity between the schools was observed in 9/11 of the cases; the same ratio was found in the Drools model, but not in the same cases.

Again, no relationship was found between the working-class preference and the mean inequity between schools:

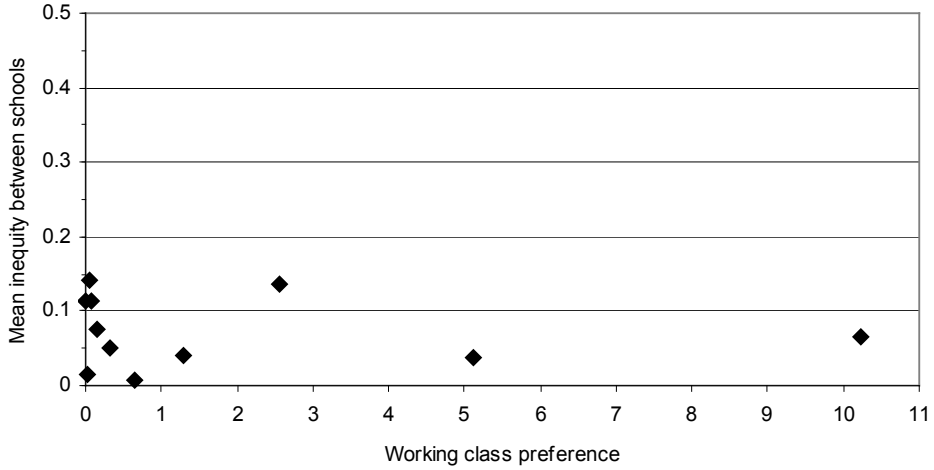


Figure 9.14 – Scatterplot illustrating the lack of relationship in the Drools model between the working-class preference and the mean inequity between the schools' middle-class intake, for two class sensitive schools and a middle-class preference of 0.99.

Since no relationship is expected between the two variables, one would not expect any relationship between the Java and the Drools model either, except that polarisation does not occur. The fact that this is the case is further support for the claim that the working-class has no impact on any aspect of the outcome (rather than it having a complex effect that is not easily apparent). However, it is a concern that the inequities found were overall lower than in the Java model, which reached 0.43 and 0.32 in two cases (see Figure 9.4).

b) Middle-class preference of 1.64

This preference was identified in the Java model as one that is guaranteed to produce polarisation. The Java model coupled this middle-class preference with the working-class preferences 0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64, 1.28, 2.56, 5.12 and 10.24, and found no significant relationship between the working-class preference and the years taken to reach total polarisation. The Drools test replicated these findings fully. The Java model found that polarisation was reached on average after 40.5 years, with a high standard deviation of 9.6. Repeating one of the test cases while varying the random seed 11 times instead of the working-class preference resulted in a mean of 32.4 years, with a standard deviation of 11. The Drools result was extremely similar to this latter result: it found a mean of 32.8, with a standard deviation of 11. Therefore, the Drools behaviour appears to be correct in this experiment, and to have a very similar level of variability as previously.

c) Middle-class preference of 11.24

Finally, the extreme case of 11.24 was re-run. Being very far from the borderline case, very little variance was exhibited, as expected. As before, total polarisation occurred during year 5 in every case. Again, the trajectory differed only slightly in the first five years: a low standard deviation of 0.0023 was found among the mean intakes of the leading schools, centred on a mean of 0.727. This exhibits an almost identically low variability as the Java model does, which has a standard deviation of 0.0049; however, the mean is somewhat higher than the Java model's mean of 0.691, which is significant given the very low variance from the means within each model's runs. Thus, the test was not entirely successful: while the time taken to polarise was approximately the same, the Drools model took a slightly steeper trajectory to get to this state. However, this slightly steeper path cannot be very significant, since it only made itself noticeable in this borderline test: if it were a serious problem, then one would expect the more 'normal' test

cases to have exhibited polarisation occurring sooner than in the Java model, but this was not found.

Sweep of the middle-class fraction of the population

The same parameter space was investigated as originally in the Java model in experiment 5.2.1, varying the fraction of the population that is middle-class. This allowed a direct comparison between results:

Theta	1/(2*theta) - 0.16		1/(2*theta) - 0.01		1/(2*theta)		1/(2*theta) + 0.64		1/(2*theta) + 10.24	
	Java	Drools	Java	Drools	Java	Drools	Java	Drools	Java	Drools
0.1	/	/	/	/	1038	/	105	/	19	10
0.2	/	/	/	/	1358	1181	77	53	7	7
0.3	/	/	/	/	527	/	92	35	5	4
0.4	/	/	/	/	unstable	/	30	26	10	5
0.5	/	/	/	/	/	/	43	37	5	5
0.6	/	/	/	/	unstable	/	23	23	4	4
0.7	/	/	/	/	/	/	15	14	3	3
0.8	/	/	/	/	unstable	unstable	15	20	3	4
0.9	/	/	/	unstable	unstable	unstable	9 (blips)	14 (blips)	3	4

Table 9.1 – Comparison of the number of years taken to polarisation in the Java model, and in the Drools model. Each row represents a different fraction of the population that is middle-class (θ); while the column pairs show the middle-class preference, where 1/(2θ) is the critical value as described in section 5.2.1. '/' means that no polarisation occurred, and 'unstable' means that polarisation was reached for at least four consecutive years, but was then lost again. Table 5.1 compares the Java model's results against the results predicted by Room and Britton.

In the middle column pair, where the middle-class preference for high-performing schools is equal to the critical value, the predicted behaviour is essentially undefined, so it seems acceptable to find this type of discrepancy there, especially when the outcome is so varied within the Java model alone. The unstable polarisation reached for the middle-class population fraction of 0.9 for a middle-class preference of the critical value - 0.01 is, however, more of a cause for concern. A plot of the run itself sheds some light on the cause:

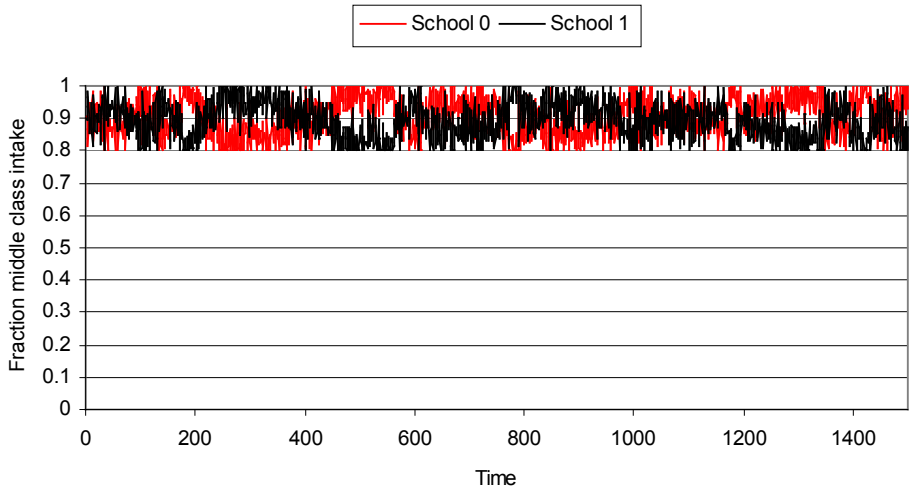


Figure 9.15 – Drools model run of a middle-class preference of the critical value - 0.01, using a middle-class fraction of the population of 0.9. 'Unstable polarisation' was reached in two instances: years 780-784, and years 787-790. Also note that from year 609 to year 610, there is a

complete role reversal, with the top school going from a value of 1.0 to the bottom fraction of 0.8 from one year to the next, showing how unstable the system is.

Because our definition of what counts as an unstable polarisation (at least four consecutive polarised time steps) is relatively lenient, this is relatively likely to occur in this type of situation, where the 1.0 mark is much more easily reached than in other cases, simply because there is such a small bracket, 0.8-1.0, in which to move. It seems that this extreme type of situation requires a different definition; perhaps it would be better to define different measures depending on the context, for future experiments.

Since it seems there is not much difference between this case of unstable 'polarisation' and the Java model's lack of polarisation, it is argued that this discrepancy is not evidence that the Drools model fails to replicate the Java model sufficiently well. The Drools model appears to be stricter about avoiding polarisation when the middle-class preference is exactly the critical value, but with such a small sample, the results are inconclusive. Certainly, the models appear to agree closely with the two higher values of middle-class preference.

9.4 Multiple schools

The following figures show a comparison of multiple-school scenarios ranging from five to ten schools.

Class sensitive schools

Most cases are depicted once with a working-class preference equal to the middle-class preference, and then with a minimal working-class preference of 0.01, to illustrate the additional complexity added by the low preference.

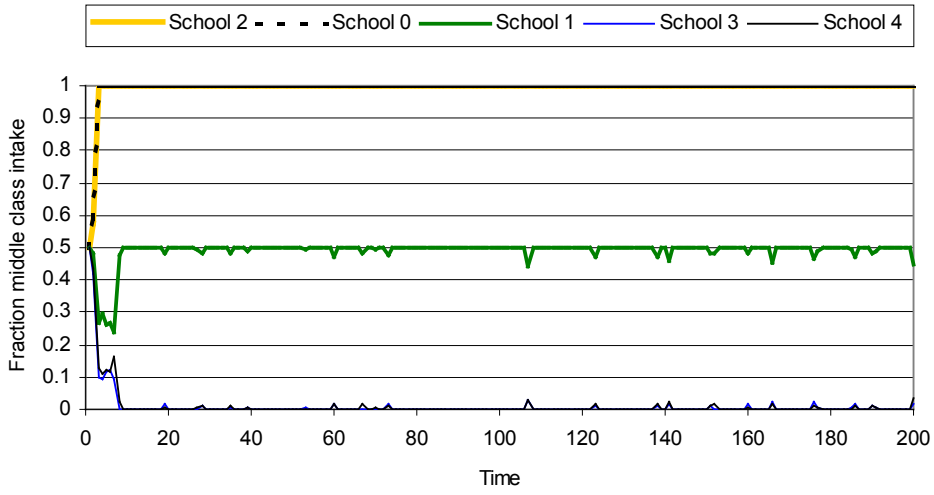


Figure 9.16 – Five class sensitive schools, with an equal middle- and working-class preference for middle-class schools of 20.48

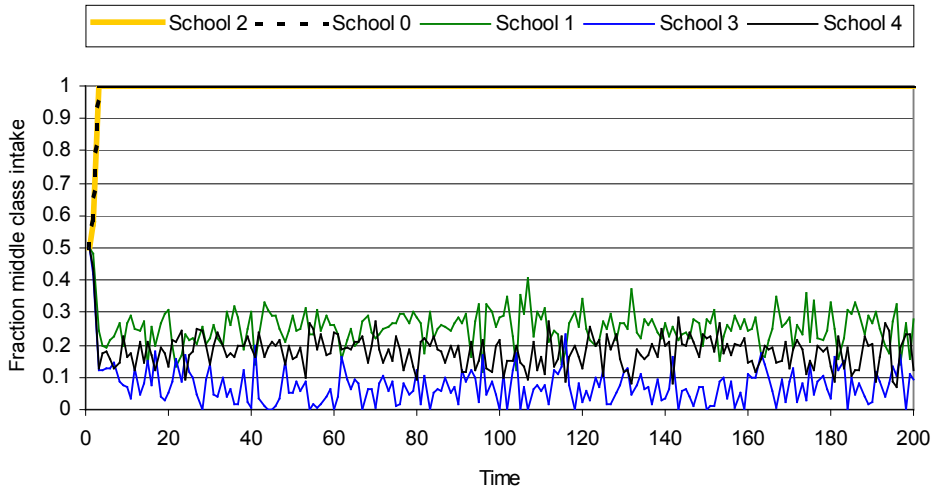


Figure 9.17 – Five class sensitive schools, with a middle-class preference for middle-class schools of 20.48, and a working-class preference of 0.01

The five-school scenario is discussed in section 6.2.

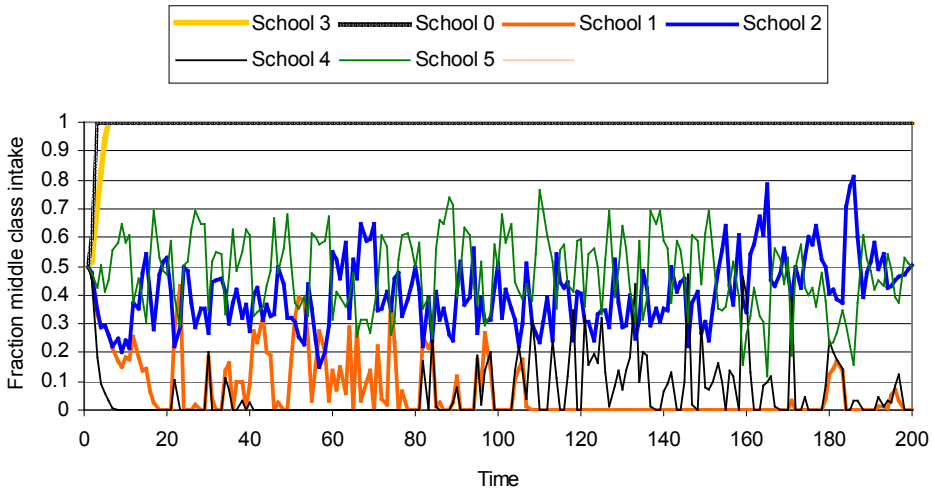


Figure 9.18 – Six class sensitive schools, with an equal middle- and working-class preference for middle-class schools of 20.48. This is essentially a more complicated version of the four-school scenario, as discussed in section 6.2.

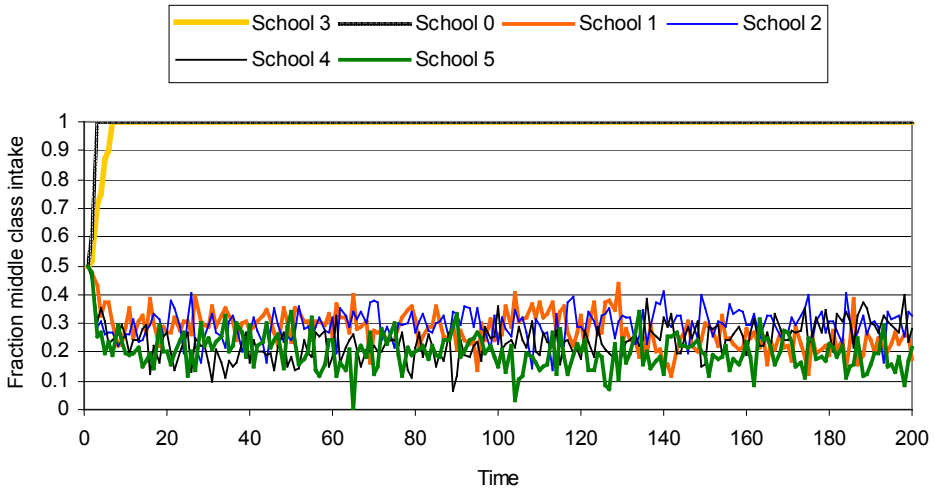


Figure 9.19 - Six class sensitive schools, with a middle-class preference of 20.48, and a working-class preference of 0.01. The additional randomness resulting from the more or less arbitrary choices by working-class families makes it more difficult for the non-leading schools to distinguish themselves above the rest of the pack than in Figure 9.18.

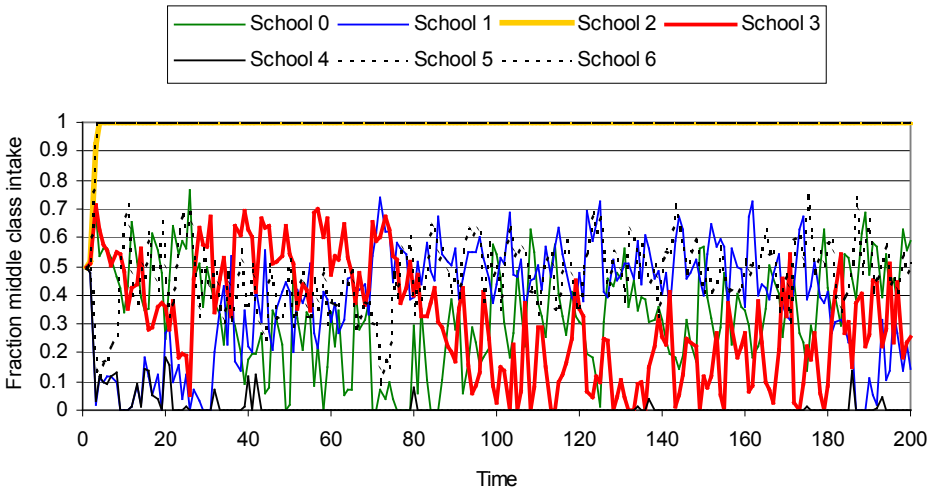


Figure 9.20 - Seven class sensitive schools, with an equal middle- and working-class preference for middle-class schools of 20.48. School 3's fall from popularity (along with related fluctuations in other schools) is worth noting; this is the greatest instability we have observed so far in these multi-school scenarios. It is, however, not surprising, given how little separated it from other schools to begin with.

Eight schools are not shown because the pattern is so similar to that of six schools; similarly, nine schools are of little interest because they follow the seven-school pattern shown above. The ten-school scenario also adds nothing new, but is shown below for completeness. Twenty schools are again similar, with six schools in the 1.0 position, rather than the ten-school scenario's four leading schools. This is interesting because the pattern is clearly not simply the ten-school scenario 'doubled'.

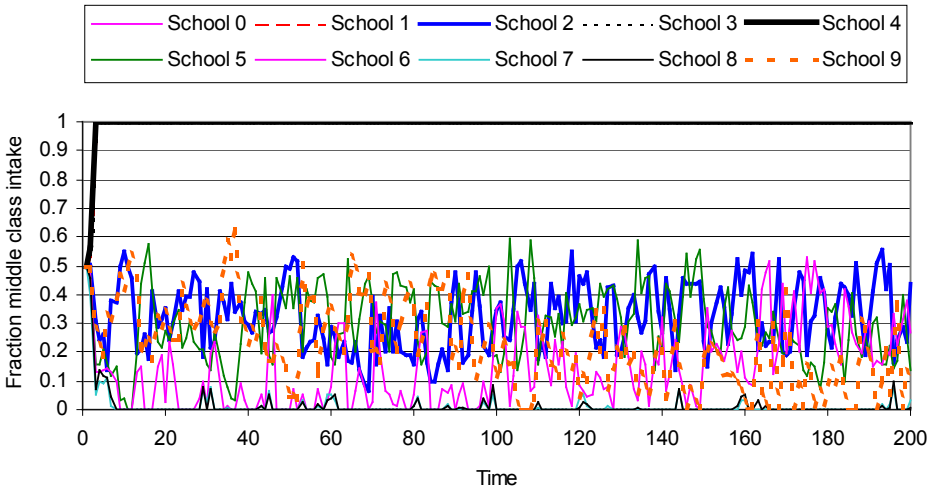


Figure 9.21 – Ten class sensitive schools, with both middle-class and working-class preferences set to 20.48. Four schools are superimposed on the 1.0 line.

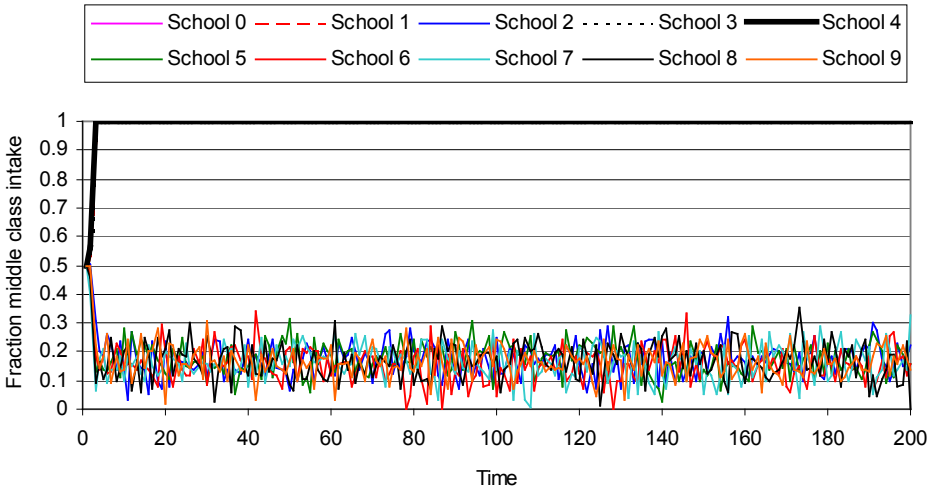


Figure 9.22 - Ten class sensitive schools, with a middle-class preference of 20.48, but a working-class preference of 0.01. Four schools are again superimposed on the 1.0 line, but the remainder of schools have had so much 'randomness' introduced by the working-class families' arbitrary choices that no differentiation can emerge.

Class blind schools

Increasing numbers of class blind schools are shown below, each with a middle-class preference of 20.48 and a working-class preference of 0.01, in order to maximise the differentiation between schools.

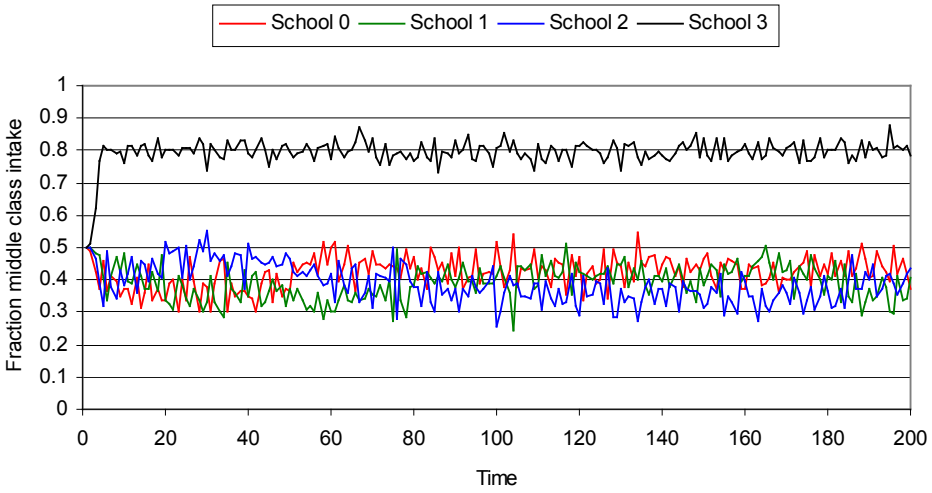


Figure 9.23 – Four class blind schools, with a middle-class preference of 20.48 and a working-class preference of 0.01

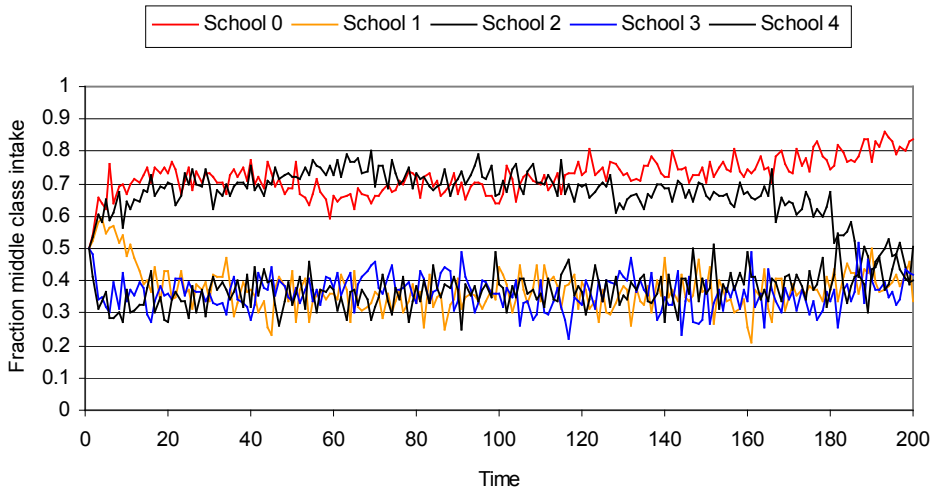


Figure 9.24 – Five class blind schools, with a middle-class preference of 20.48 and a working-class preference of 0.01

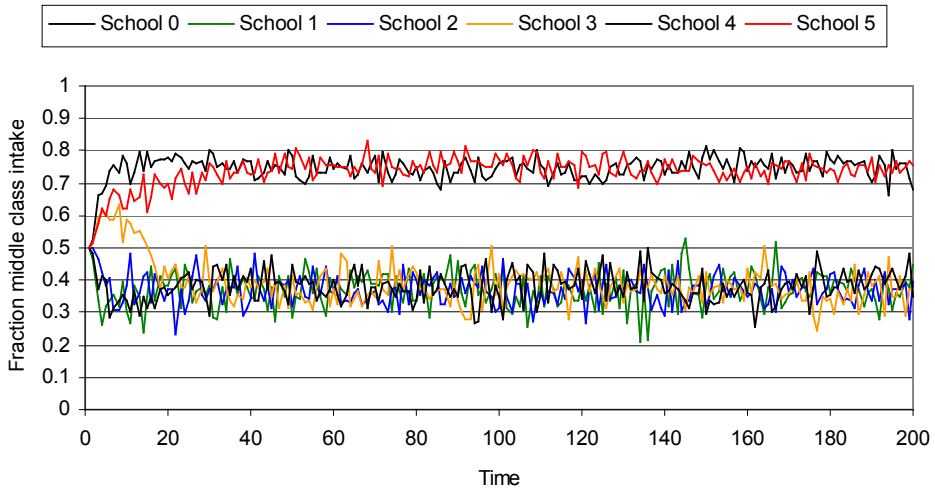


Figure 9.25 - Six class blind schools, with a middle-class preference of 20.48 and a working-class preference of 0.01

9.5 Co-evolutionary behaviour

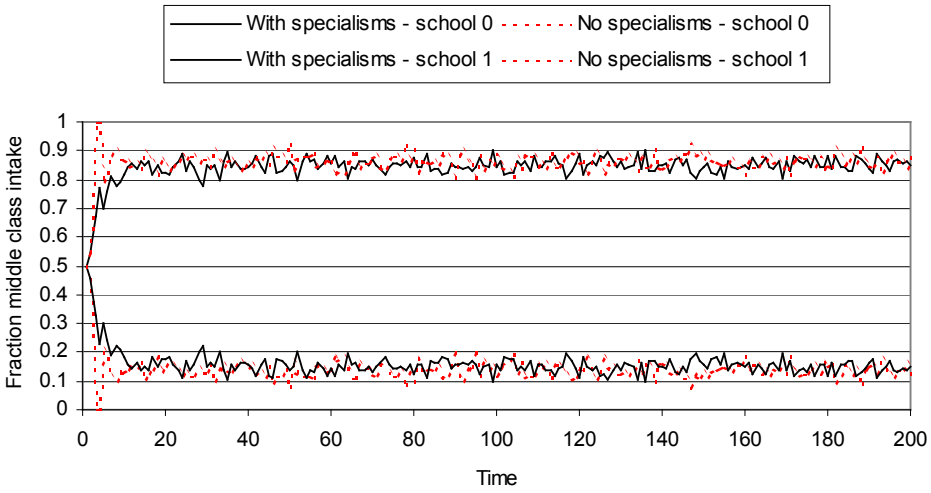


Figure 9.26 – Two runs superimposed. The simulation is initialised with two class blind schools, and a mean class blind preference of 0.32 (standard deviation 0.2). In dotted red, the only rules are the ones introduced up to experiment 6.4.5, i.e. all rules relating to schools changing their admissions strategy. In black, these rules are used as well, but school specialisms have been added to the mix as well. This softens the effect of school strategy and league table position.

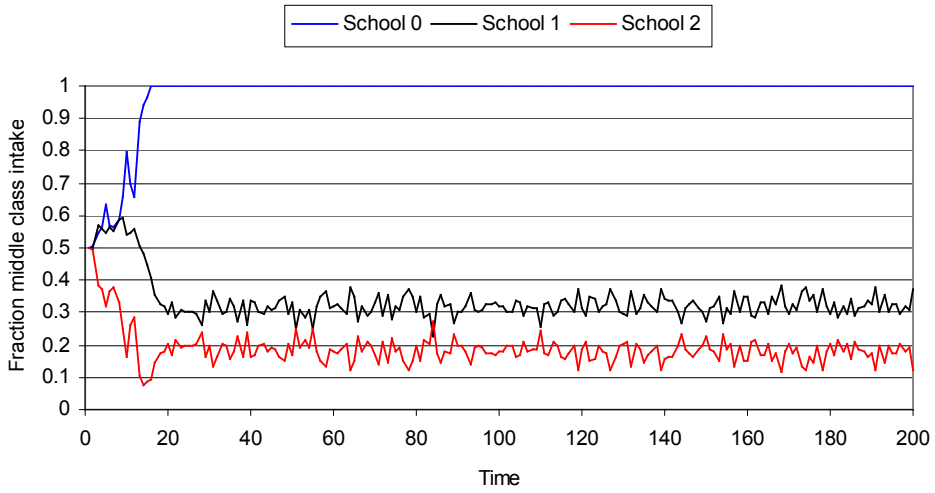


Figure 9.27 – Three schools run using the rules of experiment 6.4.4. Initially the three schools are class blind, and middle- and working-class preferences are 20.48 and 0.01 respectively. The mean class blind preference is 0.16, with a standard deviation of 0.2.

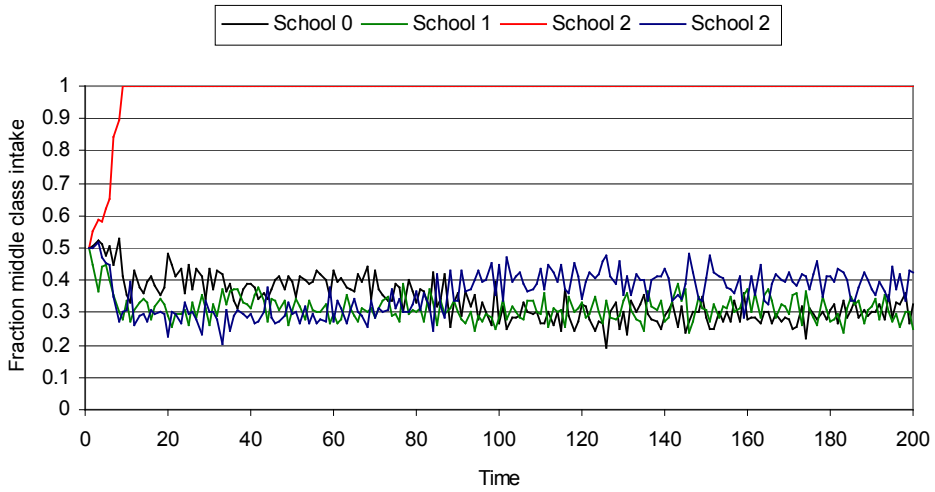


Figure 9.28 - Four schools run using the rules of experiment 6.4.4. Initially all schools are class blind, and middle- and working-class preferences are 20.48 and 0.01 respectively. The mean class blind preference is 0.32 (unlike Figure 9.27), with a standard deviation of 0.2.

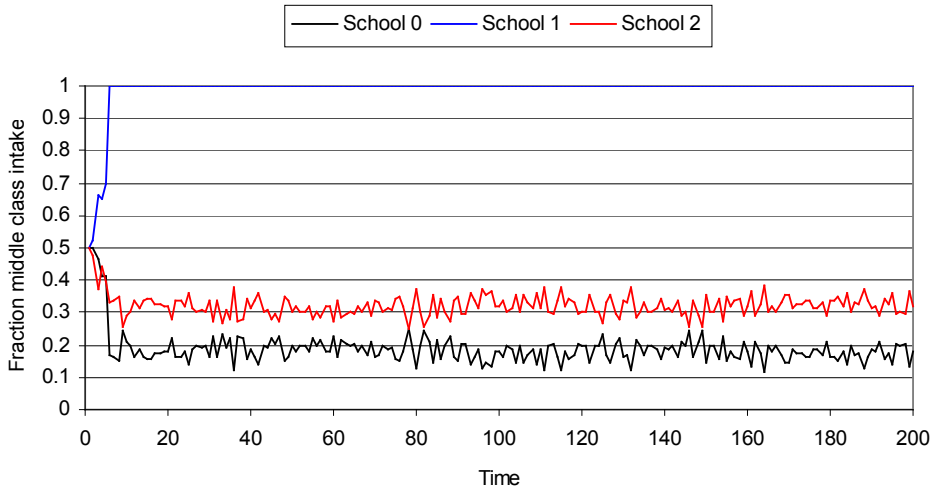


Figure 9.29 – Three schools run using the rules of experiment 6.4.9, i.e. all rules, and the parameters of Figure 9.27.

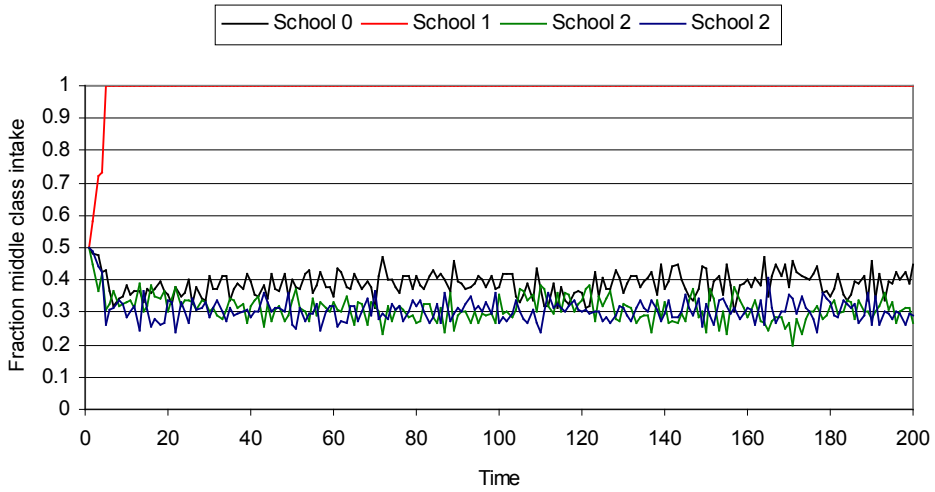


Figure 9.30 – Four schools run using the rules of experiment 6.4.9, i.e. all rules, and the parameters of Figure 9.28.

10 Appendix B

10.1 How to run the model

The model requires the Java 1.5 run-time environment to be installed.

Assuming that a JAR file named schoolChoice.jar has been generated from the source code, the following directory structure should be used:

drools-lib[33]

- drools-compiler-3.0.5.jar
- drools-core-3.0.5.jar
- drools-decisiontables-3.0.5.jar
- drools-jsr94-3.0.5.jar
- *lib*
  - antlr-2.7.6.jar
  - antlr-3.0ea8.jar
  - commons-collections-3.1.jar
  - commons-io-1.1.jar
  - commons-jci-core-1.0-406301.jar
  - commons-jci-eclipse-3.2.0.666.jar
  - commons-jci-janino-2.4.3.jar
  - commons-lang-2.1.jar
  - commons-logging-api-1.0.4.jar
  - core-3.2.0.666.jar
  - janino-2.4.3.jar
  - jsr94-1.1.jar
  - junit-3.8.1.jar
  - jxl-2.4.2.jar
  - stringtemplate-2.3b6.jar
  - xml-apis-1.0.b2.jar
  - xpp3-1.1.3.4.O.jar
  - xstream-1.1.3.jar

lib[34]

- asm.jar
- beanbowl.jar
- colt.jar
- commons-collections.jar
- commons-logging.jar
- geotools_repast.jar
- ibis.jar
- jakarta-poi.jar
- jep-2.24.jar
- jgap.jar
- jh.jar
- jmf.jar
- jode-1.1.2-pre1.jar
- joone.jar

[33] Drools library files – download from http://labs.jboss.com/jbossrules/downloads (accessed 30 April 2007)
[34] RepastJ library files – download from http://repast.sourceforge.net/download.html (accessed 30 April 2007)

JTS.jar
junit.jar
log4j-1.2.8.jar
OpenForecast-0.4.0.jar
openmap.jar
plot.jar
ProActive.jar
trove.jar
violinstrings-1.0.2.jar

repast.jar[35]

schoolChoice.jar (the model)

SchoolChoiceParams.txt (Repast parameter file – only necessary for batch mode)

Then, the mammoth command

java -cp schoolChoice.jar;repast.jar;drools-lib\drools-jsr94-3.0.5.jar;drools-lib\drools-compiler-3.0.5.jar;drools-lib\drools-core-3.0.5.jar;drools-lib\drools-decisiontables-3.0.5.jar;drools-lib\lib\antlr-2.7.6.jar;drools-lib\lib\antlr-3.0ea8.jar;drools-lib\lib\commons-collections-3.1.jar;drools-lib\lib\commons-io-1.1.jar;drools-lib\lib\commons-jci-core-1.0-406301.jar;drools-lib\lib\commons-jci-eclipse-3.2.0.666.jar;drools-lib\lib\commons-jci-janino-2.4.3.jar;drools-lib\lib\commons-lang-2.1.jar;drools-lib\lib\commons-logging-api-1.0.4.jar;drools-lib\lib\core-3.2.0.666.jar;drools-lib\lib\janino-2.4.3.jar;drools-lib\lib\jsr94-1.1.jar;drools-lib\lib\junit-3.8.1.jar;drools-lib\lib\jxl-2.4.2.jar;drools-lib\lib\stringtemplate-2.3b6.jar;drools-lib\lib\xml-apis-1.0.b2.jar;drools-lib\lib\xpp3-1.1.3.4.O.jar;drools-lib\lib\xstream-1.1.3.jar uk.ac.bath.cs.schoolchoice.SchoolChoiceModelBatch

runs the batch model using the parameter file, and replacing 'uk.ac.bath.cs.schoolchoice.SchoolChoiceModelBatch' with 'uk.ac.bath.cs.schoolchoice.SchoolChoiceModelInteractive' runs the model in interactive GUI mode.

Drools requires every library JAR file to be specified on the classpath, but Repast expects to find its library classes in the *lib* subdirectory so this is not necessary.

The old Java model can be run in an identical manner, or the *drools-lib* subdirectory can be left out and the command shortened to

java -cp schoolChoice.jar;repast.jar uk.ac.bath.cs.schoolchoice.SchoolChoiceModelBatch

or to run the GUI model,

java -cp schoolChoice.jar;repast.jar uk.ac.bath.cs.schoolchoice.SchoolChoiceModelInteractive as before.

An example parameter file is shown below. It iterates through the six combinations:

NumSpecialisms: 2

1. NumSchoolsClassBlind: 0
2. NumSchoolClassBlind: 1
3. NumSchoolsClassBlind: 2

NumSpecialisms 3

4. NumSchoolsClassBlind: 0
5. NumSchoolClassBlind: 1
6. NumSchoolsClassBlind: 2

[35] RepastJ framework itself – download from http://repast.sourceforge.net/download.html (accessed 30 April 2007)

while the other parameters remain fixed. The Repast parameter tutorial (http://repast.sourceforge.net/how-to/params.html) gives more detail on parameter file formats.

```
runs: 1
NumSpecialisms {
  set_list: 2 3
  {
    runs: 1
    NumSchoolsClassBlind {
      set_list: 0 1 2
    }
  }
}
ClassBlindPreferenceMean {
  set: 0.32
}
ClassBlindPreferenceStdDeviation {
  set: 0.2
}
MiddleClassPreference {
  set: 20.48
}
FractionMiddleClassPopulation {
  set: 0.5
}
WorkingClassPreference {
 set: 0.01
}
RngSeed {
  set: 1171139146239
}
NumChildren {
  set: 400
}
NumSchools {
  set: 2
}
NumSteps {
  set: 200
}
NumStochasticShocks {
  set: 0
}
Coevolutionary {
  set_boolean: true
}
```

10.2 Java model's actions on each step

```
shuffleChildren();
for (int i = 0; i < childList.size(); i++) {
	Child child = childList.get(i);
	shuffleSchools();
	child.applyForPlace(schoolList);
}
/*
 * Later schools' rejections can add applications back into the
 * queues of earlier schools
 */
while (existPendingApplications()) {
	for (int i = 0; i < schoolList.size(); i++) {
		School school = schoolList.get(i);
		school.processApplications();
	}
}
/*
 * Only advance the year once all applications have been
 * resolved
 */
for (int i = 0; i < schoolList.size(); i++) {
	School school = schoolList.get(i);
	school.advanceYear();
}
```

10.3 Drools audit logging

A snippet of the output produced by the Drools working memory logger is shown below, to illustrate that it is not very human-readable:

```
<object-stream>
  <list>
    <org.drools.audit.event.ActivationLogEvent>
      <activationId>Decrease preference for class blind schools due to
typical distribution [2, 1]</activationId>
      <rule>Decrease preference for class blind schools due to typical
distribution</rule>
      <declarations>fracMiddleClass=0.5(1);
applicant=uk.ac.bath.cs.schoolchoice.child.EvolutionaryChild@1458dcb(2);
env=uk.ac.bath.cs.schoolchoice.Environment@116318b(1)</declarations>
      <type>4</type>
    </org.drools.audit.event.ActivationLogEvent>
    <org.drools.audit.event.ActivationLogEvent>
      <activationId>Remove class blind factor if no schools are class
blind [0, 2]</activationId>
      <rule>Remove class blind factor if no schools are class
blind</rule>
<declarations>applicant=uk.ac.bath.cs.schoolchoice.child.EvolutionaryChil
d@1458dcb(2)</declarations>
      <type>4</type>
    </org.drools.audit.event.ActivationLogEvent>
    <org.drools.audit.event.ActivationLogEvent>
      <activationId>Research school place application [2]</activationId>
      <rule>Research school place application</rule>
```

11 Appendix C

The following pages give a selection of program code for the final, Drools-based model. The program is split into packages containing classes and rule files as follows (subclasses are indented below their superclasses where possible):

```
uk.ac.bath.cs.schoolchoice
       SchoolChoiceModel.java
              SchoolChoiceModelBatch.java
              SchoolChoiceModelInteractive.java
       Agent.java
       Environment.java
       StochasticShock.java
       YearGroup.java
       SocioEconomicStatus.java
       Output.java

uk.ac.bath.cs.schoolchoice.child
       Child.java (subclass of Agent)
       child.drl
              EvolutionaryChild.java
              evolutionaryChild.drl
       SchoolFactor.java
              LeagueTableFactor.java
              SchoolStrategyFactor.java
              SpecialismFactor.java

uk.ac.bath.cs.schoolchoice.school
       School.java (subclass of Agent)
       school.drl
              EvolutionarySchool.java
              evolutionarySchool.drl
       Specialism.java

uk.ac.bath.cs.schoolchoice.ontology
       Message.java
              Application.java
              SchoolPlaceAcceptance.java
              SchoolPlaceRejection.java

uk.ac.bath.cs.schoolchoice.deterministic
       DeterministicProbabilityRule.java
```

The four Drools .drl files, and Agent.java, School.java and EvolutionarySchool.java are shown on the following pages; the remaining Java code can be found on the CD enclosed with this submission.

11.1 child.drl

```
package uk.ac.bath.cs.schoolchoice.child

import uk.ac.bath.cs.schoolchoice.*;
import uk.ac.bath.cs.schoolchoice.school.School;
import uk.ac.bath.cs.schoolchoice.ontology.*;

/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * child.drl - decision rules used by the Child class
 */

# need to do this before *every* application (i.e. after
rejections),
# before applying again
rule "Research school place application"
salience 11
        when applicant : Child( allocatedPlace == false,
                                waitingForResponse == false,
                                schoolSelectedByProbability < 0 )
        then
                applicant.researchSchoolPlace();
end


rule "Make application for school place"
salience 10
        when
                chosenSchool : School(chosenID : ID)
                applicant : Child( allocatedPlace == false,
                                   waitingForResponse == false,

                        schoolSelectedByProbability == chosenID)
        then
       // Covers both the initial application and subsequent
       // applications after rejection
        int applicationRound = applicant.getApplicationRound()+ 1;
        chosenSchool.receiveMessage(new Application(applicant,
chosenSchool, applicationRound));
        applicant.waitForResponse(applicationRound);
end

rule "Process rejection of school place application"
salience 8
        when
        child : Child()
            school : School()
            rejection : SchoolPlaceRejection(
                                childInvolved == child,
                                schoolInvolved == school)
        then
                retract(school);
                retract(rejection);
                child.stopWaitingForResponse();
end

rule "Process acceptance of school place application"
salience 8
        when
                child : Child()
// Child should match anyway since we shouldn't have been given
// someone else's message, but check just in case
            acceptance : SchoolPlaceAcceptance(
                                childInvolved == child)
        then
                retract(acceptance);
                child.setAllocatedPlace(true);
                child.stopWaitingForResponse();
end

# Allow Child access to list of possible schools
query "all possible schools"
        school : School()
end

rule "Clean up schools once child has a place"
salience 2
        when
            Child(allocatedPlace == true)
```

```
           school : School()
        then
               retract(school);
end

rule "Catch messages that were not understood"
salience 0
        when msg : Message()
        then
               System.err.println("child: received message I do
not understand");
               retract(msg);
end
```

11.2 evolutionaryChild.drl

```
package uk.ac.bath.cs.schoolchoice.child

import uk.ac.bath.cs.schoolchoice.*;
import uk.ac.bath.cs.schoolchoice.school.EvolutionarySchool;
import uk.ac.bath.cs.schoolchoice.school.School;
import uk.ac.bath.cs.schoolchoice.ontology.*;

/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * child.drl - decision rules used by the Child class
 */

rule "Increase preference for class blind schools due to
polarisation"
# Note that this rule will fire for all children at the same time,
# since it does not depend on any attribute of Child
salience 12
        when
         applicant : EvolutionaryChild( applicationRound == 0,
                                        ignoreClassBlind == false )
         env : Environment(fracMiddleClass : fractionMiddleClass)
// Extreme polarisation detected: one school has 170% of middle
// class fraction you would normally expect. E.g. if population
// middle class fraction is 0.5, that school has >= 0.85.
// But only take action if that school is not already class blind
// - if it is, then there's no point in increasing our own
// preference for class blind schools as a mechanism for reducing
// polarisation.
         exists EvolutionarySchool(leagueTableRating >=
                                   (1.7 * fracMiddleClass),
                                preferenceForMiddleClass ==
                                School.ABSOLUTE_PREFERENCE_FOR_MIDDLE)
        then
// Only print a notification for the first child - we happen to
// know that children are numbered after schools
     if (applicant.getID() == env.getSchools().size())
     {
        Output.println("children increasing preference for class
blind schools this year");
     }
     applicant.increasePreferenceForClassBlind();
end

rule "Decrease preference for class blind schools due to typical
distribution"
# Note that this rule will fire for all children at the same time,
# since it does not depend on any attribute of Child
salience 12
        when
         applicant : EvolutionaryChild( applicationRound == 0,
                                        ignoreClassBlind == false )
         env : Environment(fracMiddleClass : fractionMiddleClass)
// All schools' middle class intake is relatively close to the
// average population middle class fraction (within 15%)
         not EvolutionarySchool(leagueTableRating >
                                   (1.15 * fracMiddleClass))
         not EvolutionarySchool(leagueTableRating <
                                   (0.85 * fracMiddleClass))
        then
                 // Only print a notification for the first child -
we happen to know that children
                 // are numbered after schools
              if (applicant.getID() == env.getSchools().size())
              {
```

```
            Output.println("children DEcreasing preference for
class blind schools this year");
        }
        applicant.decreasePreferenceForClassBlind();
end

rule "Remove class blind factor if all schools are class blind"
# Instead, place more emphasis on the other factors to allow
greater differentiation
# to emerge there
salience 13
    when
        not EvolutionarySchool(
              preferenceForMiddleClass != School.CLASS_BLIND)
        // Need this so that the rule does not keep firing
        applicant : EvolutionaryChild(applicationRound == 0,
                                      ignoreClassBlind == false)
    then
// Only print a notification for the first child - we happen to
// know that children are numbered after schools
     if (applicant.getID() ==
Environment.getInstance().getSchools().size())
     {
Output.println("Ignoring class blind factor - all class blind");
     }
     applicant.ignoreClassBlindFactor();
end

rule "Remove class blind factor if no schools are class blind"
# Instead, place more emphasis on the other factors to allow
# greater differentiation to emerge there
salience 13
    when
        not EvolutionarySchool(
              preferenceForMiddleClass == School.CLASS_BLIND)
        // Need this so that the rule does not keep firing
        applicant : EvolutionaryChild(applicationRound == 0,
                                  ignoreClassBlind == false)
    then
// Only print a notification for the first child - we happen to
// know that children are numbered after schools
    if (applicant.getID() ==
Environment.getInstance().getSchools().size())
    {
Output.println("Ignoring class blind factor - none class blind");
        }
        applicant.ignoreClassBlindFactor();
end


rule "Remove specialism factor if all schools have the same
specialism"
# Instead, place more emphasis on the other factors to allow
greater differentiation
# to emerge there
salience 13
    when
        EvolutionarySchool(allocatedSpecialism : specialism)
        not EvolutionarySchool(specialism != allocatedSpecialism)
        // Need this so that the rule does not keep firing
        applicant : EvolutionaryChild(applicationRound == 0,
                                      ignoreSpecialism == false)
    then
// Only print a notification for the first child - we happen to
// know that children are numbered after schools
    if (applicant.getID() ==
Environment.getInstance().getSchools().size())
    {
        Output.println("Ignoring specialism factor");
    }
    applicant.ignoreSpecialismFactor();
end
```

11.3 school.drl

```
package uk.ac.bath.cs.schoolchoice.school

import org.drools.QueryResults;

import uk.ac.bath.cs.schoolchoice.*;
import uk.ac.bath.cs.schoolchoice.child.Child;
import uk.ac.bath.cs.schoolchoice.ontology.*;

/**
```

```
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * school.drl - decision rules used by the School class
 */

# global variables
global java.lang.Integer myID;

# Only do this once per application phase (i.e. set of
# applications), not once per choice like Child has to
rule "Research school place application"
salience 11
        when
         school : School( placesLeft > 0,
                          applicantSelectedByProbability < 0 )

         // we have at least one pending application for the
         // correct application round
         Application(appRound : applicationRound,
                                schoolInvolved == school)
         exists Environment(applicationRound == appRound)

         // Do not consider applications yet if not all children
          // have even applied for a place yet
                not Child(allocatedPlace == false,
                          waitingForResponse == false)

        then
                school.researchApplicants();
end

rule "Accept school place application"
salience 10
        when
         chosenApplicant : Child( chosenID : ID)
         school : School( placesLeft > 0,
         applicantSelectedByProbability == chosenID )

         // Do not consider applications yet if not all children
         // have even applied for a place yet
         not Child( allocatedPlace == false,
                    waitingForResponse == false )

         // Don't need to check application round because School is
          // only given access to applicants from current
          // application round
         application : Application(
                                childInvolved == chosenApplicant,
                                schoolInvolved == school)
        then
                school.acceptApplication(chosenApplicant);
                chosenApplicant.receiveMessage(new
                SchoolPlaceAcceptance(chosenApplicant, school));
                retract(chosenApplicant);
                retract(application);

                if (school.getPlacesLeft() > 0){
                        school.resetApplicationList();
                }

// Inform competitors that they might want to recheck their
// conditions since actions have taken place
  for (School competitor : Environment.getInstance().getSchools())
        {
                if (competitor.getID() != myID)
                {
                        competitor.fireAllRules();
                }
        }
end

rule "Reject applicants for whom we have no space"
salience 13
        when
                school : School( placesLeft == 0 )
                application: Application(schoolInvolved == school)
        then
                retract(application);

                Child applicant = application.getChildInvolved();
                applicant.receiveMessage(
                new SchoolPlaceRejection(applicant, school));

// Inform competitors that they might want to recheck their
```

```
// conditions since actions have taken place
  for (School competitor : Environment.getInstance().getSchools())
        {
               if (competitor.getID() != myID)
               {
                      competitor.fireAllRules();
               }
        }
end

rule "End application round"
no-loop true
        when
               Environment(currentRound : applicationRound)
               school : School(applicationRound == currentRound)
               not Application(applicationRound == currentRound,
                               schoolInvolved == school)
// Only advance application round once all activity
// for that round has ceased - could still be awaiting
// applications for the current round otherwise
               exists Child(allocatedPlace == false)
               not Child( allocatedPlace == false,
                          waitingForResponse == false)
// Wait for all children to get a response to the current round of
// applications
               not Child (allocatedPlace == false,
                          waitingForResponse == true,
                          applicationRound == currentRound)

// Since we are not connecting the school to an application we
// have received (thereby implicitly checking that it's 'us'),
// need to explicitly check we are not looking at a competitor
// instead. Sadly cannot reference this as part of condition above
// - this is very inefficient.
            eval(school.getID() == myID)
        then
        if (school.getPlacesLeft() > 0) {
        Environment.getInstance().endApplicationRound(school);
        } else {
// No places left - alert the environment we will not be
// taking part in any further rounds this year.
               Environment.getInstance().endApplications(school);
        }
end

# Can't have 'did not understand msg' since we need them to hang
# around indefinitely

# Allow School access to list of pending applications for the
current
# round of applications only - applicants from round x must be
given precedence
# over those from round x + 1
query "all pending applicants"
  school : School()
  Environment(currentRound : applicationRound)
  application : Application(applicationRound == currentRound,
                            schoolInvolved == school,
                            applicant : childInvolved)
end
```

11.4 evolutionarySchool.drl

```
package uk.ac.bath.cs.schoolchoice.school

import org.drools.QueryResults;

import java.util.ArrayList;
import uk.ac.bath.cs.schoolchoice.*;
import uk.ac.bath.cs.schoolchoice.child.Child;
import uk.ac.bath.cs.schoolchoice.ontology.*;

/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * child.drl - decision rules used by the Child class
 */

# global variables
global java.lang.Integer myID;

// Are the *majority* of successful competitors to this school
```

```
// class sensitive? Can't do this via constraints because concept
// of 'majority' is not well supported by Drools 3.0
function boolean successfulCompetitorsMostlyClassSensitive(
        School school)
{
    int numSensitive = 0;
    int numBlind = 0;
    ArrayList<School> competitors =
Environment.getInstance().getSchools();
    double fractionMiddleClass =
Environment.getInstance().getFractionMiddleClass();
    for (School competitor : competitors)
    {
        if (school.getID() != competitor.getID() &&
   competitor.getLeagueTableRating() >= 1.1 * fractionMiddleClass)
        {
          if (competitor.getPreferenceForMiddleClass() ==
        School.CLASS_BLIND)
          {
                numBlind++;
          } else
          {
                numSensitive++;
          }
        }
    }
    return numSensitive > numBlind;
}

# Re-evaluate strategy at the end of each year
rule "Adopt class blind niche"
salience 12
activation-group "strategy switch"
        when
         env : Environment(fracMiddleClass : fractionMiddleClass)
         // This year's applications are now over
         school : EvolutionarySchool(placesLeft == 0,
                leagueTableRating < (0.9 * fracMiddleClass),
                preferenceForMiddleClass != School.CLASS_BLIND)
          eval(school.getID() == myID)

          eval(successfulCompetitorsMostlyClassSensitive(school))
        then
                Output.println(school.getID() + " switching to
                class blind to adopt niche");
                school.setSchoolStrategy(School.CLASS_BLIND);
end

rule "Popular enough to become class sensitive"
salience 12
activation-group "strategy switch"
        when
         env : Environment(fracMiddleClass : fractionMiddleClas
         // Finished this year's applications; performing well in
         // the league table which we have defined to be 10% or
         // more above an 'average' rating
         school : EvolutionarySchool(placesLeft == 0,
                leagueTableRating > (1.1 * fracMiddleClass),
                preferenceForMiddleClass == School.CLASS_BLIND,
                id : ID)
// Only switch to class sensitive if we would not be giving up
// a niche position in the class blind market; otherwise we would
// be giving up what might be our reason for success
         (
// Either we have no successful competitors
         not EvolutionarySchool(leagueTableRating >
                (1.1 * fracMiddleClass), ID != id)
         or
// Or at least one successful competitor is already class blind
         exists EvolutionarySchool(
                   leagueTableRating > (1.1 * fracMiddleClass),
                   preferenceForMiddleClass == School.CLASS_BLIND,
                   ID != id)
                  )
          eval(school.getID() == myID)
        then
         Output.println(school.getID() + " switching to class
sensitive because popular");
         school.setSchoolStrategy(
                School.ABSOLUTE_PREFERENCE_FOR_MIDDLE);
end

rule "Everyone else is class blind and we are all doing roughly
the same"

# so become class sensitive in the hope of gaining the edge in the
```

```
# current round of applications. This is a gamble because this
# will instantly make us everyone's least preferred school unless
# we manage to raise our league table performance straight away,
# through the luck of having more middle class applicants than
# average.

# If more than one school is using this rule, randomness decides
# who gets there first - only the first will end up executing it.
salience 12
activation-group "strategy switch"
        when
         env : Environment(fracMiddleClass : fractionMiddleClass)
         // Finished this year's applications; performing well in
         // the league tables which we have defined to be 10% or
         // more above an 'average' rating
         school : EvolutionarySchool(placesLeft == 0,
                preferenceForMiddleClass == School.CLASS_BLIND,
                id : ID)
         not EvolutionarySchool(
               preferenceForMiddleClass != School.CLASS_BLIND)
         not EvolutionarySchool(leagueTableRating >
                (1.1 * fracMiddleClass), ID != id)
         not EvolutionarySchool(leagueTableRating <
                (0.9 * fracMiddleClass), ID != id)

         eval(school.getID() == myID)
        then
         Output.println(school.getID() + " switching to class
sensitive because all others blind");
         school.setSchoolStrategy(
               School.ABSOLUTE_PREFERENCE_FOR_MIDDLE);
end

rule "Change specialism to gain edge over leading school"
salience 12
activation-group "specialism switch"
        when
            newSpecialism : Specialism()
            // This year's applications are now over
             school : EvolutionarySchool(placesLeft == 0,
                       yearsSinceLastSpecialismChange > 10,
                       myRating : leagueTableRating,
                       mySpecialism : specialism ->
                                    (mySpecialism != newSpecialism))

// A more successful competitor has the same specialism as us,
// so it's not doing us any good
          exists EvolutionarySchool(leagueTableRating > myRating,
                                    specialism == mySpecialism)
// No-one much better than us has this new specialism we want to
// switch to
           not EvolutionarySchool(leagueTableRating >
               (1.1 * myRating),
                specialism == newSpecialism)
          eval(school.getID() == myID)
        then
           school.changeSpecialism(newSpecialism);
end

rule "Change specialism to that of lesser school to take away its
edge"
salience 12
activation-group "specialism switch"
        when
          exists Specialism() // if specialisms are being used
          // This year's applications are now over
          school : EvolutionarySchool(placesLeft == 0,
                       yearsSinceLastSpecialismChange > 10,
                       myRating : leagueTableRating,
                       mySpecialism : specialism)
// A less successful competitor has a unique specialism -
// so we want to take it away from them! Then they will no longer
// have an advantage, and we lose nothing.
         EvolutionarySchool(leagueTableRating < (1.1 * myRating),
                            newSpecialism : specialism ->
                          (newSpecialism != mySpecialism))
// Check no-one better than us has this new specialism we want to
// switch to
         not EvolutionarySchool(leagueTableRating >
               (1.1 * myRating),
              specialism == newSpecialism)
         eval(school.getID() == myID)
        then
          school.changeSpecialism(newSpecialism);
end
```

11.5 Agent.java

```
package uk.ac.bath.cs.schoolchoice;

import java.io.IOException;

import org.drools.FactHandle;
import org.drools.QueryResults;
import org.drools.RuleBase;
import org.drools.WorkingMemory;
import org.drools.audit.WorkingMemoryFileLogger;
import org.drools.compiler.DroolsParserException;
import org.drools.compiler.PackageBuilder;
import org.drools.compiler.PackageBuilderConfiguration;

import uchicago.src.sim.engine.Stepable;
import uk.ac.bath.cs.schoolchoice.ontology.Message;

/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * Agent.java - Superclass for any agent in this simulation;
 * handles the details of an agent's Drools rule system. An agent
 * is an independent entity that makes its own decisions about
 * actions to take.
 */
public abstract class Agent {

	// Counter used to give each agent a unique ID
	private static int idCounter = 0;

	// Unique ID for this agent
	private int id;

	// Agent's own copy of rules and facts it knows of
	private WorkingMemory workingMemory;

	// Drools configuration object
	private PackageBuilderConfiguration conf;

//private WorkingMemoryFileLogger droolsLogger; - Too cumbersome

	/**
	 * Create a new agent
	 */
	public Agent() {
		id = idCounter++;
// We need to do this in order to be compatible with Java 1.5
		conf = new PackageBuilderConfiguration();
		conf.setJavaLanguageLevel("1.5");

		// Create a fresh working memory
		try {
			workingMemory =
				getRuleBase().newWorkingMemory();
		} catch (Exception e) {
			System.err.println("Unable to load Drools
					rulebase");
			e.printStackTrace();
		}
		// Monitor the environment for changes
		assertVariable(Environment.getInstance());

//droolsLogger = new WorkingMemoryFileLogger( workingMemory );
	}

	/**
	 * Initialise this agent's knowledge base. This must be
	 * called after all constructor initialisation has been
	 * done - not before.
	 */
	protected void initRules() {
		workingMemory.assertObject(this, true);
	}

//	protected void audit()
//	{
//		//	writes to event.log
//		droolsLogger.writeToDisk();
//	}

	/**
	 * Get the rulebase used by this agent
```

```
     *
     * @return RuleBase
     * @throws Exception
     *     if there is a problem finding or loading the rules
     */
    protected abstract RuleBase getRuleBase()
            throws Exception;

    /**
     * Tells the agent to check for rule activations and run
     * them. It will not do this spontaneously, but it will
     * keep executing any new rules that become activated
     * through its actions, until none are triggered.
     */
    public void fireAllRules() {
            workingMemory.fireAllRules();
    }

    /**
     * Remove the given fact from working memory
     *
     * @param handle - reference given by working memory when
     * the fact was asserted
     */
    protected void retract(FactHandle handle) {
            workingMemory.retractObject(handle);
    }

    /**
     * Assert the given object into working memory as a fact;
     * tell Drools to subscribe to the object's property
     * change events in case its properties change. The object
     * must provide Java bean-like property listener methods,
     * as described in section 1.6.4.5 of the Drools manual,
     * at http://labs.jboss.com/file-access/default/members/
     * jbossrules/freezone/docs/3.0.5/html/
     * ch01s06.html#d0e708. (It is recommended to use
     * oreilly.hcj.references.PropertyChangeSupport rather
     * than java.beans.PropertyChangeSupport; it does not
     * suffer from memory leaks.)
     *
     * @param obj - fact to assert
     * @return - reference to the fact; this can be used to
     *         retract it at a later point
     */
    protected FactHandle assertVariable(Object obj) {
            return workingMemory.assertObject(obj, true);
    }

    /**
     * Assert the given object into memory as a constant fact;
     * Drools will  assume that it is immutable, and will not
     * check it for changes.
     * @param obj - fact to assert
     * @return - reference to the fact; this can be used to
     *         retract it at a later point
     */
    protected FactHandle assertConstant(Object obj) {
            return workingMemory.assertObject(obj, false);
    }

    /**
     * Set the value of the global variable of the given name
     * in working memory. The global must have been previously
     * defined in the rule file. Note that Drools assumes
     * globals are immutable.
     * @param name -
     *            name of the global variable
     * @param value -
     *            value to assign the variable
     */
    protected void setGlobal(String name, Object value) {
            workingMemory.setGlobal(name, value);
    }

    /**
     * Get the results of running the query of the given name
     * in this agent's working memory.
     *
     * This method unfortunately had to be made public to give
     * the rule engine used by this agent access, but no other
     * class should use it.
     * @param query -
     *            name of the query
     * @return results of running it
     */
```

```
public QueryResults getQueryResults(String query) {
	return workingMemory.getQueryResults(query);
}

/**
 * @return unique agent ID
 */
public int getID() {
	return id;
}

/**
 * This method can be called once on this class at the end
 * of a simulation run, in order to reset the agent
 * identifiers to restart at 0 for the next run. This
 * makes the output more legible, but is not strictly
 * necessary.
 */
public static void resetNumbering() {
	idCounter = 0;
}

/**
 * Receive a message sent by another agent, and process it
 * @param msg - message
 */
public void receiveMessage(Message msg) {
	assertConstant(msg);
	fireAllRules();
}

/**
 * Create a new PackageBuilder with the correct
 * configuration, for loading packages into working
 * memory.
 * @return PackageBuilder
 * @throws DroolsParserException
 * @throws IOException
 */
protected PackageBuilder getPackageBuilder()
	throws DroolsParserException,
		IOException {
	return new PackageBuilder(conf);
	}

	/**
	 * Clean up resources when this agent is finished with.
	 * The agent will no longer function after this method is
	 * called.
	 */
	public void dispose() {
		workingMemory.dispose();
		workingMemory = null;
	}
}
```

11.6 School.java

```
package uk.ac.bath.cs.schoolchoice.school;

import uk.ac.bath.cs.schoolchoice.*;

import java.beans.PropertyChangeListener;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.Iterator;

// Use memory safe version - java.beans one is not!
import oreilly.hcj.references.PropertyChangeSupport;

import org.drools.FactHandle;
import org.drools.QueryResult;
import org.drools.QueryResults;
import org.drools.RuleBase;
import org.drools.RuleBaseFactory;
import org.drools.compiler.DroolsParserException;
import org.drools.compiler.PackageBuilder;
import org.drools.rule.Package;

import uchicago.src.collection.RangeMap;
import uchicago.src.sim.engine.Stepable;
```

```
import uchicago.src.sim.util.Random;
import uk.ac.bath.cs.schoolchoice.child.Child;
import uk.ac.bath.cs.schoolchoice.deterministic.
              DeterministicProbabilityRule;

/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * School.java - represents a school.
 *
 * It uses the Accessor design pattern, so that Drools can
 * automatically
 * subscribe to its PropertyChangeListeners rather than us having
 * to tell Drools explicitly every time an attribute changes.
 */
public class School extends Agent implements Stepable {
// May need to make this variable in future, if schools can open
// partway through the simulation
       private final static int yearOpened = 0;
// School strategy of random allocation of applicants to places
       public static final double CLASS_BLIND = 1;
// School strategy of unconditionally favouring middle-class
// applicants over working-class ones.
// double.MAX_VALUE is too high for our libary probability code to
// cope with
       public static final double ABSOLUTE_PREFERENCE_FOR_MIDDLE
= Integer.MAX_VALUE;

// The preference the school has for middle-class students
// (CLASS_BLIND, ABSOLUTE_PREFERENCE_FOR_MIDDLE or somewhere
// in between)
       protected double preferenceForMiddleClass = CLASS_BLIND;

// The number of places in the school for each yeargroup
       private int numPlaces;

// The current year
       protected int year = 0;

// The current application round this year
       protected int applicationRound = 0;

// The fraction of middle-class children this school was
// initialised with.
       private double initialFractionMiddleClass;

// The number of years in a school, i.e. how many yeargroups
// there are
       private static final int numYears = 6;

// The number of years a school's league table rating is
// calculated over
       private static final int leagueTableYears = 4;

// The current pending, partially filled yeargroup
       private YearGroup pendingYearGroup;

// List of the this year's applicants, so we can remove them from
// working memory at the end of the year
       private ArrayList<FactHandle> currentApplicants = new
ArrayList<FactHandle>();

// Random number used to select the next applicant to accept
// from the probability map
       private double randomNumberForApplicants = -1;

// List of ratings given to applicants, from which a probability
// map can be generated
       private DeterministicProbabilityRule childProbabilities;

// Probability map of applicants and how likely we are to
// select them
       private RangeMap applicantPreferenceMap = null;

// Any stochastic shocks that have been applied to the school
       private ArrayList<StochasticShock> stochasticShocks =
               new ArrayList<StochasticShock>();

// Yeargroups, hashed by year of entry to the school
       private Hashtable<Integer, YearGroup> yearGroups;

// Helper class to inform anyone who has registered an interest
// that a property of this school has changed
       protected final PropertyChangeSupport changes =
```

```
                new PropertyChangeSupport( this );

// Rule base of compiled Drools rules the school uses to make its
// decisions. Shared by all schools for efficiency.
        private static RuleBase ruleBase = null;

// Utility class for building up a rule base
        private static PackageBuilder packageBuilder = null;

        /**
         * Create a new school
         *
         * @param preferenceForMiddleClass - how much the school
         * prefers middle class students, i.e. its admissions
         * policy.
         * Possible values include CLASS_BLIND and
         * ABSOLUTE_PREFERENCE_FOR_MIDDLE.
         * @param numPlaces - number of places in school each year
         * @param initialFractionMiddleClass - initial fraction of
         * middle-class children in the school's composition, at
         * the start of the simulation
         */
        public School(double preferenceForMiddleClass,
                      int numPlaces,
                       double initialFractionMiddleClass )
        {
                super();
                this.preferenceForMiddleClass =
                preferenceForMiddleClass;
                this.numPlaces = numPlaces;
                this.initialFractionMiddleClass =
                initialFractionMiddleClass;
                setup();
        }

        /**
         * Create a new school
         *
         * @param preferenceForMiddleClass - how much the school
         * prefers middle class students, i.e. its admissions
         * policy.
         * Possible values include CLASS_BLIND and
         * ABSOLUTE_PREFERENCE_FOR_MIDDLE.
         * @param numPlaces - number of places in school each year
         */
        public School(double preferenceForMiddleClass,
                      int numPlaces)
        {
                super();
                this.preferenceForMiddleClass =
                preferenceForMiddleClass;
                this.numPlaces = numPlaces;
                initialFractionMiddleClass =
                Environment.getInstance().getFractionMiddleClass();
                setup();
        }

        /**
         * Set up the school, finishing the creation process.
         * Must be called in the constructor.
         */
        private void setup()
        {
                yearGroups = new Hashtable<Integer, YearGroup>();
                pendingYearGroup = new YearGroup(yearOpened,
                                                 numPlaces);

                initRules();
        }

        /**
         * Initialise the agent's rules
         */
        protected void initRules()
        {
                setGlobal("myID", getID());
                super.initRules();
        }

        /**
         * Inform the school that all its competitors are set up,
         * so it can initialise fully.
         */
        public void initialise()
        {
// Nothing to do - don't care that competitors are set up too
```

```
}

/**
 * @return the school's preference for middle-class
 * applicants, aka its strategy
 */
public double getPreferenceForMiddleClass()
{
	return preferenceForMiddleClass;
}

/**
 * @return the pending, partially filled yeargroup for the
 * upcoming year
 */
public YearGroup getPendingYearGroup()
{
	return pendingYearGroup;
}

/**
 * @return the number of places per year in this school
 */
public int getNumPlaces() {
	return numPlaces;
}

/**
 * @return the number of places left in the current year
 * in this school
 */
public int getPlacesLeft() {
	if (pendingYearGroup != null)
	{
		return numPlaces -
		pendingYearGroup.getNumChildren();
	} else
	{
		return numPlaces;
	}
}

/**
 * @return the current application round
 */
public int getApplicationRound() {
	return applicationRound;
}

/**
 * @param appRound - the current application round
 */
public void setApplicationRound(int appRound) {
	int oldState = applicationRound;
	applicationRound = appRound;
	int newState = applicationRound;
	this.changes.firePropertyChange("applicationRound",
					oldState,
					newState);
}

/**
 * Register the given property change listener, adding it
 * to the list of objects that wish to be informed of any
 * change to this school's properties.
 * @param l - listener to add
 */
public void addPropertyChangeListener(
	final PropertyChangeListener l) {
	this.changes.addPropertyChangeListener( l );
}

/**
 * Remove the given property change listener from the
 * list of objects subscribed to this school's changes.
 * @param l - listener to de-register
 */
public void removePropertyChangeListener(
	final PropertyChangeListener l) {
	this.changes.removePropertyChangeListener( l );
}

/**
 * Get the school's league table rating - this is a moving
 * average over the past four years.
 */
```

```
        public double getLeagueTableRating()
        {
                double total =
                getFractionMiddleClassChildren(year);
                double fraction;
                int count = 1;

                while (count <= leagueTableYears)
                {
                        fraction =
                getFractionMiddleClassChildren(year - count);
                        if (fraction >= 0 )
                        {
                                total += fraction;
                        } else
                        {
// We do not yet have enough history to look at past 4 years
                                break;
                        }
                        count++;
                }

                double rating = total / count;

                // Take account of all stochastic shocks received
                for (int i = 0; i < stochasticShocks.size(); i++)
                {
                        // May be a positive or a negative effect
                        rating += stochasticShocks.get(i).
                                getStochasticImpactAt(year);
                }

                // This is supposed to be a fraction, so truncate
                // any excesses due to stochastic shocks
                if (rating > 1)
                {
                        rating = 1;
                }
                if (rating < 0)
                {
                        rating = 0;
                }
                        return rating;
        }

        /**
         * @param calcYear - year to investigate
         * @return fraction of middle class kids in the entire
         * school as it was during the given year
         */
        private double getFractionMiddleClassChildren(
                                                int calcYear)
        {
                if (calcYear == yearOpened) /* no history */
                {
                        return initialFractionMiddleClass;
                }

                if (calcYear > year)
                {
// cannot answer question regarding the future

                        return -1;
                }

                int totalChildren = 0;
                int middleClassChildren = 0;

                YearGroup group;
                for (int i = yearGroups.size() - 1 -
                                (year - calcYear);
             // only look at students who haven't yet graduated
                     i >=0 && i >= yearGroups.size() - numYears;
                     i--)
                {
                        group = yearGroups.get(i);
                        totalChildren += group.getNumChildren();
                        middleClassChildren +=
                        group.getNumChildren(
                SocioEconomicStatus.socialClass.MIDDLE_CLASS);
                }

                return (double)middleClassChildren /
                                (double)totalChildren;
        }
```

```
/**
 * @return fraction of middle class children in the entire
 * school
 */
public double getFractionMiddleClassChildren()
{
       double frac = getFractionMiddleClassChildren(year);
       return frac;
}

/**
 * @param socClass - social class to investigate
 * @return number of children in the school of the given
 * social class
 */
public int getNumberChildrenOfClass(
              SocioEconomicStatus.socialClass socClass)
{
       int children = 0;

       YearGroup group;
       for (int i = yearGroups.size() - 1;
    // only look at students who haven't yet graduated
            i >=0 && i >= yearGroups.size() - numYears;
            i--)
       {
              group = yearGroups.get(i);
              children += group.getNumChildren(socClass);
       }
       return children;
}

/**
 * @return the number of children in the school, of any
 * social class
 */
public int getNumberAnyClass()
{
       int children = 0;

       YearGroup group;
       for (int i = yearGroups.size() - 1;
    // only look at students who haven't yet graduated
            i >=0 && i >= yearGroups.size() - numYears;
            i--)
       {
              group = yearGroups.get(i);
              children += group.getNumChildren();
       }
       return children;
}

/**
 * @return number middle-class children in the entire
 * school
 */
public int getNumberMiddleClass()
{
       return getNumberChildrenOfClass(
          SocioEconomicStatus.socialClass.MIDDLE_CLASS);
}

/**
 * @return number working-class children in the entire
 * school
 */
public int getNumberWorkingClass()
{
       return getNumberChildrenOfClass(
          SocioEconomicStatus.socialClass.WORKING_CLASS);
}

/**
 * @return fraction of middle-class children accepted in
 * the most recent year
 */
public double getFractionMiddleClassIntake()
{
       if (year == yearOpened) /* no history */
       {
              return initialFractionMiddleClass;
       }

       YearGroup group = yearGroups.get(
              yearGroups.size() - 1);
```

```
        int totalChildren = group.getNumChildren();
        int middleClassChildren = group.getNumChildren(
           SocioEconomicStatus.socialClass.MIDDLE_CLASS);

        return (double)middleClassChildren /
                        (double)totalChildren;
}

/**
 * Accept the given child's application for a school place
 * Note: this method should only ever be called by Drools;
 * it is unfortunate that it needs to be public.
 * @param child
 */
public void acceptApplication(Child child)
{
        int oldStatePlaces = getPlacesLeft();
        int oldStateChosen =
        getApplicantSelectedByProbability();

        try {
                pendingYearGroup.add(child);
        } catch (Exception e) {
                e.printStackTrace();
        }

        childProbabilities.removeNode(child);
        applicantPreferenceMap = null;
        randomNumberForApplicants = -1;

        int newStatePlaces = getPlacesLeft();
        int newStateChosen =
        getApplicantSelectedByProbability();
        this.changes.firePropertyChange("placesLeft",
                                        oldStatePlaces,
                                        newStatePlaces);
        this.changes.firePropertyChange(
                "applicantSelectedByProbability",
                oldStateChosen,
                newStateChosen);
}

/**
 * Get the preference rating given by the school to the
 * given child
 * @param child
 * @return rating/preference for the child
 */
private Double getPreferenceRating(Child child)
{
        double basePreference = 1;
        if (child.getSocialClass() ==
        SocioEconomicStatus.socialClass.MIDDLE_CLASS)
        {
                return preferenceForMiddleClass *
                           basePreference;
        } else
        {
                return basePreference;
        }
}

/**
 * The school is affected by a stochastic shock in the
 * given year.
 * The stochastic shock class is responsible for setting
 * its parameters.
 */
public void takeStochasticShock()
{
        stochasticShocks.add(new StochasticShock(year));
        log("Received shock of "
        + stochasticShocks.get(stochasticShocks.size()
                        - 1).getStochasticImpactAt(year)
        + " during year " + year);
}

/**
 * Log the given message
 * @param msg - message
 */
protected void log(String msg)
{
        Output.println("School " + getID() + ": " + msg);
}
```

```
/**
 * Prepare for the up-coming admissions process
 * at the start of the new academic year.
 */
public void preStep()
{
	childProbabilities = new
	DeterministicProbabilityRule();

	int oldState = getPlacesLeft();
	pendingYearGroup = new YearGroup(year, numPlaces);
	this.changes.firePropertyChange("placesLeft",
					oldState,
					getPlacesLeft());

	ArrayList<Child> applicants =
	Environment.getInstance().getCurrentApplicants();
	currentApplicants.clear();
	FactHandle handle;
	for (Child child : applicants)
	{
		handle = assertVariable(child);
		currentApplicants.add(handle);
	}
}

/**
 * Process the children's applications for school places
 */
public void step()
{
	fireAllRules();
}

/**
 * Clean up at the end of the academic year
 */
public void postStep()
{
	yearGroups.put(year, pendingYearGroup);
	year++;

	for (FactHandle han : currentApplicants)
	{
		retract(han);
	}
}

/**
 * Research all children who have made an application to
 * this school, calculating a preference rating for each
 * and selecting one by probability distribution.
 */
public void researchApplicants()
{
	ArrayList<Child> pendingApplications =
	getPendingApplications();
// Assign each child a probability according to preference for it
	childProbabilities = new
	DeterministicProbabilityRule(pendingApplications);
	for (Child applicant : pendingApplications)
	{
		double probability =
		getPreferenceRating(applicant);
		childProbabilities.update(applicant,
			(float)probability);
	}
	resetApplicationList();
}

/**
 * @return the applicant who was last selected by
 * probability distribution
 */
public int getApplicantSelectedByProbability()
{
	if (applicantPreferenceMap == null ||
		randomNumberForApplicants < 0)
	{
		return -1;
	}
	return ((Child)applicantPreferenceMap.get(
	randomNumberForApplicants)).getID();
}

/**
```

```
 * @return list of all current applicants
 */
private ArrayList<Child> getPendingApplications()
{
        ArrayList<Child> pendingApplications =
        new ArrayList<Child>();
        QueryResults results = getQueryResults(
                "all pending applicants");
        QueryResult result;
        for (Iterator it = results.iterator();
             it.hasNext();)
        {
                result = (QueryResult)it.next();
                Child applicant =
                        (Child)result.get("applicant");
                pendingApplications.add(applicant);
        }
        return pendingApplications;
}

/**
 * A child has just been accepted, so remove it from the
 * application list and select the next one by
 * probability distribution.
 */
public void resetApplicationList()
{
        ArrayList<Child> pendingApplications =
                getPendingApplications();

        if (pendingApplications.size() > 0)
        {
                int oldState =
                        getApplicantSelectedByProbability();

                applicantPreferenceMap = new RangeMap();
                applicantPreferenceMap =
                   childProbabilities.makeProbabilityMap(
                        applicantPreferenceMap);
                randomNumberForApplicants =
                   Random.uniform.nextDoubleFromTo(0, 1);

                int newState =
                        getApplicantSelectedByProbability();
                changes.firePropertyChange(
                        "applicantSelectedByProbability",
                        oldState,
                        newState);
        }
}
/**
 * Create a package builder for use by this class, if it
 * does not already exist.
 */
private void setPackageBuilder() throws
        DroolsParserException, IOException
{
        if (packageBuilder == null)
        {
                packageBuilder = getPackageBuilder();
        }
}
/**
 * Get a package builder but do not set it for use by this
 * class
 */
protected PackageBuilder getPackageBuilder() throws
                DroolsParserException, IOException
{
        Reader source = new InputStreamReader(
                School.class.getResourceAsStream(
                "/school.drl" ) );
        PackageBuilder builder = super.getPackageBuilder();
        // this will parse and compile in one step
        builder.addPackageFromDrl( source );
        return builder;
}
/**
 * Get a set of compiled rules for this class
 */
protected RuleBase getRuleBase() throws Exception
{
        if (ruleBase == null)
        {
                setPackageBuilder();
```

```
                //get the compiled package (which is serializable)
                        Package pkg = packageBuilder.getPackage();

        //add the package to a rulebase (deploy the rule package).
                        ruleBase = RuleBaseFactory.newRuleBase();
                        ruleBase.addPackage( pkg );
                }
                return ruleBase;
        }

        /**
         * Dispose of the school's resources when we are finished
         * with it
         * @see uk.ac.bath.cs.schoolchoice.Agent#dispose()
         */
        @Override
        public void dispose() {
                super.dispose();
        // Necessary to ensure correct running of next simulation:
                // want to start from blank sheet
                packageBuilder = null;
                ruleBase = null;
        }
}
```

11.7 EvolutionarySchool.java

```
package uk.ac.bath.cs.schoolchoice.school;

import java.io.InputStreamReader;
import java.io.Reader;
import java.util.ArrayList;
import org.drools.RuleBase;
import org.drools.RuleBaseFactory;
import org.drools.compiler.PackageBuilder;
import org.drools.rule.Package;

import uchicago.src.sim.util.Random;
import uk.ac.bath.cs.schoolchoice.Environment;
import uk.ac.bath.cs.schoolchoice.Output;
/**
 * Multi-agent simulation of the dynamics of school choice
 * Perdita Robinson
 * May 2007
 *
 * EvolutionarySchool.java - Specialisation of School using
 * adaptive/co-evolutionary behaviour.
 */
public class EvolutionarySchool extends School {

        // The subject specialism (if any) adopted by the school
        private Specialism specialism = null;

        // What year this school last changed its specialism in
        private int changedSpecialism = -1;

        // Compiled rulebase used by all evolutionary schools
        private static RuleBase ruleBase = null;
        /**
         * Create a new school with evolutionary behaviour
         * @param preferenceForMiddleClass - the preference held
         * by the school for middle-class applicants. Only the
         * preferences  School.CLASS_BLIND and
         * School.ABSOLUTE_PREFERENCE_FOR_MIDDLE are supported.
         * @param numPlaces - how many places school has per year
         * @param initialFractionMiddleClass - the initial
         * fraction of middle-class children the school has
         * at the start of the simulation
         * @throws Exception on invalid input
         */
        public EvolutionarySchool(double preferenceForMiddleClass,
                                  int numPlaces,
                                  double initialFractionMiddleClass)
throws Exception {
                super(preferenceForMiddleClass,
                      numPlaces,
                      initialFractionMiddleClass);
                if (preferenceForMiddleClass != School.CLASS_BLIND
                    && preferenceForMiddleClass !=
                       School.ABSOLUTE_PREFERENCE_FOR_MIDDLE) {
                       throw new
                RepastException("EvolutionarySchool supports no
                middle class preferences other than class blind and
                fully class sensitive");
```

```
        }
        Output.println("School " + getID() + " is "
        + (preferenceForMiddleClass == School.CLASS_BLIND ?
        "" : "not ") + "class blind.");
        setup();
}
/**
 * Create a new school with evolutionary behaviour
 *
 * @param preferenceForMiddleClass - the preference held
 * by the school for middle-class applicants. Only the
 * preferences School.CLASS_BLIND and
 * School.ABSOLUTE_PREFERENCE_FOR_MIDDLE are supported.
 * @param numPlaces - how many places school has per year
 * @throws Exception on invalid input
 */
public EvolutionarySchool(double preferenceForMiddleClass,
                          int numPlaces)
throws Exception {
        super(preferenceForMiddleClass, numPlaces);
        if (preferenceForMiddleClass != School.CLASS_BLIND
                && preferenceForMiddleClass !=
        School.ABSOLUTE_PREFERENCE_FOR_MIDDLE) {
        throw new RepastException("EvolutionarySchool
        supports no middle class preferences"
        + " other than class blind and fully class
        sensitive");
        }
        Output.println("School " + getID() + " is "
        + (preferenceForMiddleClass == School.CLASS_BLIND ?
        "" : "not ") + "class blind.");
        setup();
}
/**
 * @return school's specialist subject
 */
public Specialism getSpecialism() {
        return specialism;
}

/**
 * Set up the school at the beginning of the simulation.
 */
private void setup() {
        ArrayList<Specialism> specialisms =
                Environment.getInstance().getSpecialisms();
        // Pick a specialism
        if (specialisms.size() > 0) {
                int index = Random.uniform.nextIntFromTo(0,
                        specialisms.size() - 1);
                specialism = specialisms.get(index);
                Output.println(getID() + " My specialism is
                        " + specialism.getType());
        }
        // Tell Drools about all possible specialisms
        for (Specialism spec : specialisms) {
                assertConstant(spec);
        }
}
/**
 * Change specialism to the specified one
 * @param newSpecialism - specialist subject
 */
public void changeSpecialism(Specialism newSpecialism) {
        if (!specialism.equals(newSpecialism)) {
                int oldState =
                        getYearsSinceLastSpecialismChange();
                changedSpecialism = year;
                int newState =
                        getYearsSinceLastSpecialismChange();
                this.changes.firePropertyChange(
                        "yearsSinceLastSpecialismChange",
                        oldState,
                        newState);
        }

        Specialism oldState = specialism;
        specialism = newSpecialism;
        this.changes.firePropertyChange("specialism",
                                        oldState,
                                        newSpecialism);
        Output.println(getID() + " My new specialism is " +
                specialism.getType());
}
/**
 * @return years since the school last changed its
```

```
         * specialism
         */
        public int getYearsSinceLastSpecialismChange() {
                if (changedSpecialism < 0) {
                        return year;
                }
                return year - changedSpecialism;
        }
        /**
         * Inform the school that all its competitors are set up,
         * so it can initialise fully.
         */
        public void initialise() {
        // Tell Drools about our competitors so it can monitor
        // their strategies and adapt to them
        for (School sch : Environment.getInstance().getSchools())
                        if (sch.getID() != getID()) {
                                assertVariable(sch);
                        }
                }
        }
        /**
         * Get the compiled set of rules for all evolutionary
         * schools
         * @return compiled rules
         */
        protected RuleBase getRuleBase() throws Exception {
                if (ruleBase == null) {
                        PackageBuilder builder =
                                getPackageBuilder();

                // Add extra rules onto standard School ones
                        Reader source = new InputStreamReader(
School.class.getResourceAsStream("/evolutionarySchool.drl"));
                // this will parse and compile in one step
                        builder.addPackageFromDrl(source);

                // get the compiled packages (this is fine as long
                // as they use the same namespace)
                        Package pkg = builder.getPackage();

                // add the package to a rulebase (deploy the rule
                // package)
                        ruleBase = RuleBaseFactory.newRuleBase();
                        ruleBase.addPackage(pkg);
                }
                return ruleBase;
        }
        /**
         * Set the school's strategy to the given one
         * @param strategy - new strategy
         * @throws Exception on invalid input: only
         * School.CLASS_BLIND and
         * School.ABSOLUTE_PREFERENCE_FOR_MIDDLE are allowed
         */
        public void setSchoolStrategy(double strategy)
                throws Exception {
                if (strategy == School.CLASS_BLIND
                     || strategy ==
                          School.ABSOLUTE_PREFERENCE_FOR_MIDDLE) {
                        double oldState = preferenceForMiddleClass;
                        double newState = strategy;
                        preferenceForMiddleClass = strategy;
                        changes.firePropertyChange(
                                "preferenceForMiddleClass",
                                oldState,
                                newState);
                } else {
                        throw new RepastException(
                        "Evolutionary schools are
                        only compatible with strictly "
                        + "class blind or strictly class sensitive
                        strategies.");
                }
        }

        /**
         * Dispose of the school's resources; we are finished with
         * the entire object
         */
        public void dispose() {
                super.dispose();
                ruleBase = null;
        }
}
```